智能车辆先进技术丛书

Model Predictive Control for Self-driving Vehicles

无人驾驶车辆模型预测控制

龚建伟 姜 岩 徐 威 著
陈慧岩 主审

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

策划编辑：李炳泉　李秀梅
执行编辑：靳　媛
封面设计：七星工作室

无人驾驶车辆模型预测控制

Model Predictive Control for Self-driving Vehicles

ISBN 978-7-5640-9084-5

定价：68.00元

智能车辆先进技术丛书

无人驾驶车辆模型预测控制

龚建伟　姜　岩　徐　威　著
陈慧岩　主审

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

内容提要

本书主要介绍模型预测控制理论与方法在无人驾驶车辆运动规划与跟踪控制中的应用。由于模型预测控制理论数学抽象特点明显，初涉者往往需要较长时间的探索才能真正理解和掌握，而进一步应用到具体研究，则需要更长的过程。本书详细介绍了应用模型预测控制理论进行无人驾驶车辆控制的基础方法，结合运动规划与跟踪控制实例详细说明了预测模型建立、方法优化、约束处理和反馈校正的方法，给出了 Matlab/CarSim 仿真代码和详细图解仿真步骤。所有代码都提供了详尽的注解，并且融入了研究团队在本领域的研究成果。

本书可以作为地面无人车辆、空中无人机、无人艇及移动机器人等无人车辆模型预测控制的研究参考资料，同时也可以作为学习模型预测控制理论的应用教材。

图书在版编目（CIP）数据

无人驾驶车辆模型预测控制/龚建伟，姜岩，徐威著．—北京：北京理工大学出版社，2014.4

ISBN 978-7-5640-9084-5

Ⅰ.①无…　Ⅱ.①龚…②姜…③徐…　Ⅲ.①无人驾驶-汽车-模型-预测控制　Ⅳ.①U46

中国版本图书馆 CIP 数据核字（2014）第 076602 号

出版发行 /北京理工大学出版社有限责任公司
社　　址 /北京市海淀区中关村南大街 5 号
邮　　编 /100081
电　　话 /（010）68914775（总编室）
　　82562903（教材售后服务热线）
　　68948351（其他图书服务热线）
网　　址 /http：//www.bitpress.com.cn
经　　销 /全国各地新华书店
印　　刷 /北京地大天成印务有限公司
开　　本 /710 毫米×1000 毫米　1/16
印　　张 /13
字　　数 /220 千字
版　　次 /2014 年 4 月第 1 版　2014 年 4 月第 1 次印刷
定　　价 /68.00 元

责任编辑 /梁铜华
文案编辑 /梁铜华
责任校对 /周瑞红
责任印制 /王美丽

前言

无人驾驶车辆技术在智能交通系统和军事领域有着广阔的应用前景。著者所在北京理工大学机械与车辆学院智能车辆研究所（Intelligent Vehicle Research Center, Beijing Institute of Technology）自1990年开始进行无人驾驶车辆技术研究和探索，并且于2009—2013年连续5届参加国家自然科学基金委主办的“中国智能车未来挑战赛”，取得良好成绩，其中在2013年比赛中获得冠军。

在研究过程中，我们认识到无人驾驶车辆的环境感知、规划决策方法与车辆平台控制是密切相关的，特别是高速无人车辆的运动学与动力学特性不仅影响到感知算法，而且与规划算法和跟踪控制算法有着内在的联系。车辆运动学约束与动力学约束如果不能有效体现在运动规划与控制算法、传感器感知算法中，就无法得到很好的控制效果。

模型预测控制有其天然的多模型约束处理优势，能够与规划控制、感知过程的传感器数据预处理算法很好地结合，是在无人驾驶车辆控制过程中体现车辆运动学与动力学约束的理想方法；但模型预测控制算法较为复杂，计算方法复杂，对于不是从事控制理论学习的工科学者比较抽象。我们将在无人驾驶车辆研究过程中应用模型预测控制理论的基础概念和方法、算法代码整理出来，一方面给课题组形成较为系统的学习资料，方便后续研究；另一方面也可以供从事移动机器人、无人车辆（包括无人机、无人艇）的研究者们参考。另外，也可作为车辆工程、自动控制等专业高年级本科生和研究生学习模型预测控制的辅助资料。

除封面署名作者外，实验室在读研究生刘凯、孙银健、袁盛玥、蒋健、徐大陆、张瑞琳参与了部分章节的初稿写作、代码测试和全书校对，于宏啸提供附录A说明；陈慧岩教授和熊光明副教授对全书内容进行了审校；清华大学汽车研究所李升波博士审阅了本书部分内容，给出了很中肯的修改意见。在此一并致谢。

本书的研究工作得到国家自然科学基金项目“高速地

面车辆主动危险规避最优运动规划与控制的动力学模型分析”（51275041）资助，是该项目前期的初步研究成果。后续工作将进一步在高速车辆控制过程中应用模型预测控制方法。本书的出版，也得到国家自然科学基金项目“复杂交通条件下弱环境约束区域无人驾驶车辆相对定位方法关键问题的研究”（61304194）部分资助。该项目提供了实验车辆的定位和感知方法基础，本书所述工作同时也与著者龚建伟于2011—2012年在美国麻省理工学院Karl Iagnemma博士主持的Robotic Mobility Group课题组访问研究期间从事的研究工作相关。该研究工作促成了基金项目“51275041”的申报，同时这些研究成果也被收录在本书中。

代码下载地址：http://www.bitpress.com.cn。

著　者

目录

第 1 章

无人驾驶车辆与模型预测控制

无人驾驶车辆（Self-driving Vehicle）是地面无人车辆的一种，在未来智能交通系统中有着广阔的应用前景。不仅各大汽车厂商纷纷研发无人驾驶技术测试样车，互联网与移动通信公司也积极涉足这一领域。本章首先介绍无人车辆通用概念、无人驾驶车辆与无人车辆的关系，说明无人车辆的主要组成部分，即任务决策、环境感知、路径规划、路径跟踪与车辆平台控制子系统，总结了无人驾驶车辆的特点，最后介绍模型预测控制理论在无人驾驶车辆控制过程中的应用情况。

1.1 无人驾驶车辆

1.1.1 无人车辆通用概念

无人车辆（Unmanned Vehicle），根据其行驶环境的不同，可以分为空中无人机（Unmanned Aerial Vehicle）、水面无人艇（Unmanned Surface Vehicle）、水下无人潜水器（Unmanned Underwater Vehicle）和地面无人车辆（Unmanned Ground Vehicle）。无人车辆是一种可以较高速度或高速移动的机器人，能够感知行驶环境，进行自主决策，规划行驶路径，并控制车辆跟踪期望路径，到达设定的目的地，完成预定任务。与机器人类似，无人车辆可以独立或者协调合作完成预定任务。同时，需要指出的是，遥操作也是无人车辆的一种重要控制

方式。遥操作无人车辆不等同于完全的遥控车辆，而是在自主行驶无人车辆基础上，增加了人在回路的任务决策、环境感知与规划、控制等功能，是一种提升自主无人车辆控制能力的人机交互方式，或者被称为人在回路的决策、规划与控制方法[1]。

根据以上定义，这里将无人车辆系统分为任务决策、环境感知、路径规划，以及车辆控制与平台4个子系统。

（1）任务决策子系统

无人车辆首先是一个任务平台，在军事领域，能够完成侦察、打击、排暴和后勤保障等任务，空中、水面、水下、地面的各种无人车辆之间，或者无人车辆与有人车辆之间能够进行任务分配与协调，不同功能（异构）或相同功能（同构）的车辆协同完成规定任务；在民用领域，无人车辆也可以完成农业灌溉、采矿、码头运输、交通运输等任务。多无人车辆协同完成任务时，具有更高的效率。其任务分配与协调在这里不做详细介绍，可以参阅著者前期研究文献[2，3，4]。

当任务分配到单台无人车辆后，无人车辆就需要根据任务特性、自身功能条件及已知环境信息进行任务决策与规划。无人车辆在任务决策与规划时，任务完成往往与车辆平台全局路径规划是关联的。比如军用无人侦察车辆任务规划包括两部分内容：一是根据车辆平台任务载荷特点确定完成任务的方式；二是确定完成任务的全局路径，进行初始全局路径规划。任务决策与规划是一种动态规划，需要根据任务和全局环境信息的变化进行调整。以完成侦察任务为例，无人车辆需要遍历侦察区域，同时需要规划出遍历这些区域的最短或最经济全局路径。

（2）环境感知子系统

和有人操控车辆一样，无人车辆需要实时得到行驶环境信息。环境信息一般可以通过以下两种途径获得：一是通过无人车辆环境感知子系统利用车载传感器系统获取行驶环境信息，结合环境模型对传感器信息进行融合、理解和识别行驶环境；二是通过通信网络提供的外部环境信息，例如车联网给无人驾驶车辆提供的前方道路拥堵情况和周围车辆行驶趋势，路基交通设施发送的路口交通信号灯情况和变化趋势，或者地面指挥系统给无人机提供的大范围空域环境信息。

环境感知子系统利用上述环境信息，并结合先验环境模型，就能对行驶环境进行理解识别。环境感知是无人车辆可靠行驶的关键。在各种无人车辆中，空中无人机的行驶环境识别较为简单。这也是现在无人机无论是在军事领域还是在民用领域都已经得到广泛应用的原因。地面无人车辆的环境感知是最复杂

的。以行驶在城市环境的无人驾驶车辆为例，如果不依赖外部网络提供的环境信息，独立的无人驾驶车辆需要识别行驶道路及与交通规则相关的各类环境信息，包括车道线以及无车道线情况下的道路识别[5-9]、路沿检测[10]、交通标识与信号灯检测[11,12]，以及行人、车辆检测[13]等，而且要综合多种道路交通要素对比较复杂的环境进行理解，为路径规划子系统提供行驶区域信息和障碍物信息。

（3）路径规划子系统

路径规划（Path Planning）是指在具有障碍物的环境中，按照一定的评价标准，比如路径长度最短或能量消耗最少原则等，寻找一条从起始状态到目标状态的无碰撞路径。无人车辆路径规划主要继承了机器人研究领域关于路径规划的研究成果[14-20]，一般分为全局路径规划[1,15,17,18,19,20]和局部路径规划[14,15]。全局路径规划功能同时也被视为任务决策规划的一部分。在大多数情况下，无人系统全局路径规划和任务决策与规划是相联系的，二者结合得到无人车辆的全局路径信息；而局部路径规划则是在无人车辆周围局部环境里进行的。

全局路径规划是在地图已知的情况下，利用已知局部信息，如障碍物位置和道路边界，确定可行和最优的路径；但是当环境发生变化，如出现未知障碍物时，就必须通过局部路径规划来生成无人车辆的局部行驶路径或轨迹。局部路径规划是在全局路径引导下，依据传感器感知到的局部环境信息来实时生成车辆所需要行驶的路径。在规划过程中不仅要考虑影响当前任务完成的最优原则，如路径最短、能源消耗最少，或者车辆最安全等，而且要考虑动态环境带来的约束问题。

路径规划技术的研究重点主要包括环境建模和路径搜索策略两个子问题，如图1.1所示。路径规划与障碍物环境信息密切相关，因此障碍物环境的建模方法是路径规划问题研究中的一个关键问题。目前，最常用的3种环境建模方法是单元分解法（Cell Decomposition）、道路图法（Roadmap）和人工势场法（Potential Fields）[21]。其中人工势场法在实时避障方面具有独特优势。在此基础上，又提出了虚拟力场法（Virtual Force Fields，VFF）、矢量场直方图法（Vector Field Histogram，VFH）和向量极坐标直方图法（Vector Polar Histogram，VPH）[14,22]。这些方法在机器人和无人车辆障碍规避方面得到大量应用。

在无人车辆和机器人局部路径规划过程中，还需要考虑的一个重要问题是运动规划（Motion Planning），即局部路径规划要满足无人车辆的运动学与动力学约束条件。文献[16]提出的满足无人车辆运动微分约束的纵横向协同

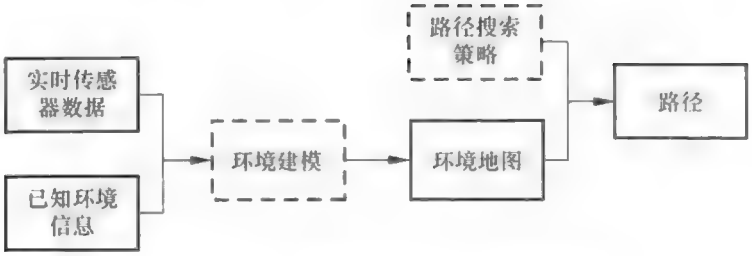


图 1.1　路径规划、环境信息与环境建模的关系

规划算法就是一种典型的局部路径规划的运动规划方法。将模型预测控制理论应用在无人车辆运动规划时，也将重点探索该方法在处理运动学和动力学约束的途径，在后续章节中将对其重点阐述。

（4）车辆平台控制子系统

车辆控制主要是控制车辆平台跟踪路径规划子系统得到的路径，也即车辆的横向与纵向控制[23-25]。和机器人控制一样，无人车辆也存在着路径（Path）与轨迹（Trajectory）的区分。本书约定轨迹是路径的一种，即同时考虑空间和时间因素的路径。路径跟踪实质是通过控制车辆的运动来减少车辆与参考路径之间的空间上的误差。如果考虑轨迹，则还包括时间误差。因模型预测控制重点应用在跟踪控制方法中，后续章节将继续对这一点进行详细说明。

车辆平台是无人车辆的重要组成部分。环境感知、决策规划及控制必须与车辆平台进行一体化设计。各种无人车辆在行驶环境中，以较高速度行驶时，都会与环境发生相互作用，这时车辆动力学与运动学特性就会影响到环境感知、决策规划和控制效果。以最为复杂的地面无人车辆为例，高速行驶的车辆执行机构的控制输入、轮胎与地面摩擦引起的滑移、横向加速度引起的侧倾等动力学非线性约束条件比低速时更加苛刻。因此，无人车辆一方面要在运动规划阶段计算出满足车辆动力学和运动学约束的无碰撞运动轨迹，另一方面需要在跟踪阶段生成满足非线性动力学约束和执行机构极限约束的控制量。

1.1.2　考虑乘坐舒适性的无人驾驶车辆

由上面简要介绍的通用无人车辆概念及其主要组成部分可以看出，无人车辆种类有很多。有的车辆有乘坐人员，但不直接驾驶；而有的没有乘坐人员，完全不需要考虑人员的乘坐感受。根据其特点不同，规划与控制时需考虑的要素也不同。如军用无人车辆在危险环境中行驶时，只需要搭载执行任务的执行

机构、传感器以及车辆平台本身能够适应环境，因此规划与控制时就只需要考虑机器的承受能力，而不需要考虑人的乘坐舒适性，这样有可能形成更高的机动能力。本书主要以与普通乘用车类似的地面无人车辆为对象进行建模，并且约定该车辆为前轮转向，即一种能够自动或自主驾驶的无人驾驶车辆（Self-driving Vehicle），虽然没有驾驶员，但还有乘客，因此需要考虑人的乘坐舒适性。

选择有人乘坐的无人驾驶车辆（Self-driving Vehicle）还有两个原因：一是这类车辆的运动学模型和动力学模型都比较成熟，在仿真软件 CarSim 和 Adams 中都有相应的模型，便于我们重点介绍模型预测控制理论在无人驾驶车辆控制中的应用；二是著者前期公开发表的工作成果也都是基于这类车辆平台，并且有多辆可以实际测试的无人驾驶车辆可作为验证平台。图 1.2 所示为北京理工大学智能车辆研究所与比亚迪汽车有限公司共同研发的 Ray 号无人驾驶车辆。

图 1.2　无人驾驶车辆 Ray（获 2013 年“中国智能车未来挑战赛”冠军）

1.2　路径跟踪与轨迹跟踪

模型预测控制理论将主要应用在无人驾驶车辆的路径或轨迹生成与跟踪控制中。这里首先介绍无人车辆路径与轨迹的概念，然后说明路径跟踪（Path Following）与轨迹跟踪（Trajectory Tracking）的特点。

1.2.1　路径规划与轨迹规划

路径与轨迹延续了机器人控制的概念。对于无人车辆来说，全局路径点只要包含空间位置信息即可，也可以包含姿态信息，而不需要与时间相关，但局

部规划时，则可以考虑时间信息。这里规定轨迹点也是一种路径点，即当路径点信息中加入了时间约束，就可以被称为轨迹点。从这个角度理解，轨迹规划就是一种路径规划，当路径规划过程要满足无人车辆的纵向和横向动力学约束时，就成为轨迹规划。路径规划与轨迹规划既可以在状态空间中表示[15]，也可以在笛卡尔坐标系中表示[16]。

无人车辆纵向速度规划与时间密切相关，而车辆横向动力学对纵向速度的影响非常大。文献［16］提出一种基于运动微分约束的无人车辆纵、横向协同规划算法，实际上考虑了无人车辆通过某一路径点的速度要求，是一种轨迹规划算法。

1.2.2 路径跟踪与轨迹跟踪

路径跟踪过程中，参考路径（Reference Path）曲线可与时间参数无关，跟踪控制时，可以假设无人车辆以当前速度匀速前进，以一定的代价规则形成行驶路径趋近于参考路径；而轨迹跟踪时，参考路径曲线与时间和空间均相关，并要求无人车辆在规定的时间内到达某一预先设定好的参考路径点。

路径跟踪不同于轨迹跟踪，不受制于时间约束，只需要在一定误差范围内跟踪期望路径。路径跟踪中的运动控制就是寻找一个有界的控制输入序列，以使无人车辆从一个初始位形到设定的期望位形。文献［26］指出，无人车辆系统是一个欠驱动的非完整约束系统，同时也是一个零动力学系统。轨迹跟踪问题会受限于这种不稳定的零动力学约束。

1.3 模型预测控制在无人驾驶车辆运动规划与控制中的应用

车辆动力学模型分析对于解决无人驾驶车辆运动规划与控制问题具有十分关键的作用。首先，在规划与控制系统中引入动力学模型可以通过模型的等效约束转化减少规划与控制的计算量，提高系统的实时性。其次，以准确的动力学模型作为预测模型，可以提高控制器对车辆未来行为的预测能力，进而在保证车辆稳定运行的同时，充分发挥车辆的机动潜能，比如高速运行状况下的主动危险规避运动规划与控制，充分体现车辆平台运动学与动力学约束条件的影响。

模型预测控制最明显的优点是能在控制过程中增加多种约束。无人车辆在低速时，车辆平台运动学约束影响较大，而随着速度的增加，动力学特性对运动规划与控制的影响就越明显，带来多种模式的约束。这正是模型预测控制在

无人车辆运动规划控制方面的应用优势。下面介绍地面无人驾驶车辆的运动规划和控制算法研究现状，同时说明模型预测控制理论在这些算法中的应用情况。

1.3.1　运动规划算法的模型约束

研究无人驾驶车辆的运动规划问题时，利用感知系统提供的障碍物位置与运动属性、路面特性以及车辆本身状态，可计算出从当前位置到局部目标位置的没有碰撞的行驶轨迹。根据前面对路径规划与轨迹规划的特点介绍，一旦轨迹被确定了，车辆沿着路径的运动形式也就被确定了。轨迹规划既可以是在路径规划后的二次规划，也可以根据感知信息一次规划获得。

（1）路径规划算法约束条件

无人驾驶车辆的局部路径规划算法不同于一般移动机器人的路径规划问题。为满足车辆转向过程中的平稳性以及安全性需求，车辆运动微分约束和动力学特性约束是需要考虑的重要特征，同时还需要考虑车辆操纵稳定性及舒适性问题带来的控制约束。文献［26］利用弹性带理论（Elastic Band Theory）对无人驾驶车辆的紧急避障路径进行规划，在规划时提出将路径的局部曲率变化最小作为约束条件，从而提高车辆紧急避障时的操纵稳定性。文献［27］讨论了非结构化环境下基于滚动窗口的无人车辆路径规划问题，并在规划过程中考虑了车辆的安全约束。

局部路径规划算法中考虑约束的一种思路是对车辆行驶曲线进行描述，通过设计评价函数选取不同工况下的行驶曲线完成规划。文献［16］利用5次多项式描述行驶曲线，将问题变为求取连接两个确定的目标点之间的曲线，并且满足目标点处有固定的航向角和固定曲率的要求，如图1.3所示；同时，在全局或者局部环境中的参考路径上用不同预瞄距离确定若干个预瞄位姿，对于每个预瞄位姿沿其法线方向按照不同横向位置偏差再确定若干个目标位姿，各目标位姿航向与预瞄位姿相同，计算当前车辆位姿和各目标位姿之间的行驶曲线作为候选曲线集合，行驶曲线需要满足车辆运

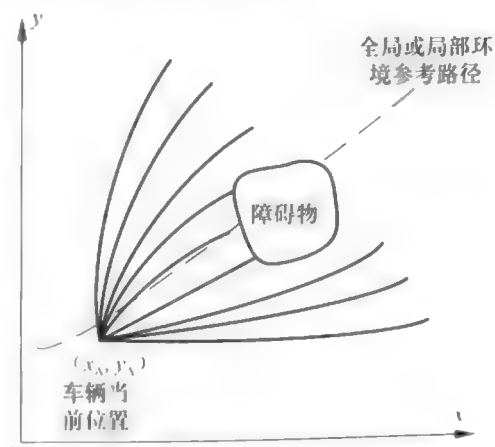


图1.3　基于5次多项式的路径规划

动微分约束。候选曲线集合包含了用不同预瞄距离生成的行驶曲线，而对于同一预瞄距离处的多条行驶曲线，其目标位姿相对于期望路径是对称分布的。首先对各条行驶曲线进行碰撞分析，在行驶曲线上的每一位姿处增加车辆的宽度和长度信息，与构形空间中的障碍物检测结果进行对比，判断行驶曲线与障碍物发生碰撞的位置，仅保留此位置以前的行驶曲线。然后，在具有相同横向位置偏差的行驶曲线中保留最长的一条用于路径评价。经过碰撞安全性分析，在每个横向位置偏差仅保留了一条满足横向安全和路径跟踪要求的行驶曲线。车辆的实际驾驶行为，如跟随或超车，则由选择哪一条行驶曲线作为待执行路径实现。

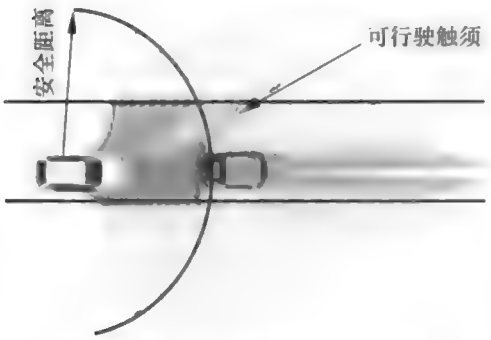


图 1.4　基于触须的路径规划

德国卡尔斯鲁厄大学在 DARPA Urban Challenge 的参赛车辆 Annie-WAY 上采用了基于触须原理的局部路径规划算法[28]，利用多组不同半径的圆弧描述不同车速下的行驶路径，如图 1.4 所示。文献［18］对该算法进行了改进，使之适用于全局环境未知的智能车自主行驶，但由于它在进行触须生成时采用圆弧曲线，而忽略了车辆转弯半径的变化，故在车速较快时容易发生跟踪失败的情况。这也说明在高速时还需要进一步考虑车辆动力学特性的约束。

文献［29］结合预测控制理论进行无人车辆的局部路径规划，借鉴预测控制的滚动优化原理，提出了基于滚动窗口的移动机器人路径规划方法。该方法在全局环境未知时能够充分利用已探知的局部信息，有效地结合优化与反馈机制，不仅使局部规划的计算量保持在较低水平，而且保证了全局的收敛性，获得了较好的规划效果。文献［30］同样借鉴了滚动优化的原理，提出了一种带约束曲线拟合的路径规划方式，如图 1.5 所示。该方法通过在规划中引入几何约束，如起始位姿约束、最小转弯半径约束等，有效地避免了在跟踪过程中同时考虑位置和航向误差的问题。

（2）轨迹规划算法约束条件

轨迹规划问题一般是在考虑系统某项性能最优的同时，加入边界约束、环境几何约束以及系统的动力学约束等非线性约束，最终构成一个非线性最优化问题。因此，轨迹规划的主要研究方向集中在非线性最优求解和约束条件等价转换上。轨迹规划问题一般无法找到显式解，普遍的做法就是将这一优化问题转换为非线性规划问题，进而通过相应的非线性规划求解器求解。

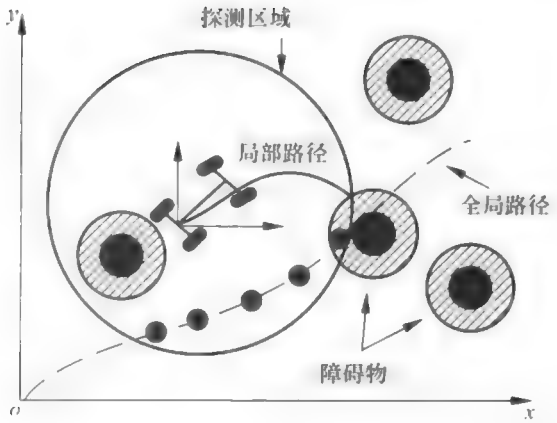


图 1.5　带约束曲线拟合的路径规划

文献 [31] 采用直接法，利用多项式直接对系统状态进行参数化，进而将轨迹规划问题转换为数学规划问题。文献 [32] 在借鉴样条法原理的基础上，通过局部近似的方法构建全局连续的优化轨迹。虽然这种转换解决了在轨迹空间优化的难题，但是其优化的速度依赖于参数空间的维度。为此，Fahroo 等人在对非线性系统平坦性研究的基础上，给出了平坦系统的轨迹规划方法[33]。其主要优点就是通过系统的平坦性将高维度空间的参数优化问题降为低维度参数空间的优化问题，从而提高了优化速度。吉林大学陈虹等也应用微分平坦理论对不同形式的机器人进行了轨迹规划方面的研究[34,35]。

侧倾约束下的运动规划也是高速运动车辆需要考虑的一个重要约束。侧倾主要采用横向载荷转移率（LTR）、侧倾角和横向加速度作为判断标准。Tahirovic[36] 在被动侧翻控制基础上进行了模型预测改进运动规划；Iagnemma[52,63] 在机器人车辆运动轨迹规划时，设计车辆动力学参数值窗口，采用速度－曲率轨迹方法计算运动轨迹。由于采用窗口值只考虑曲率，所以其计算速度快，但并不具备车辆控制稳定的最优性。

在约束条件处理方面，基于简化运动学模型满足车辆非完整约束的方法比较普遍[1,35]。假设车辆在运动过程中没有滑移，那么就是非完整（Nonholonomic）动力学约束，但是这种方法不适用于高速行驶车辆，因为此时车辆的约束已经不再局限于非完整约束，车辆的横向动力学成为规划中必须考虑的因素。文献 [39] 和 [40] 在考虑车辆动力学约束情况下对车辆的行驶轨迹进行了规划，实现了连续曲率的轨迹规划。

为了减少动力学约束的复杂程度，忽略车辆尺寸信息简化而来的点质量模型也在轨迹规划中得到了应用[41-43]。其受力分析如图 1.6 所示。从图 1.6 中

可以看出，该模型考虑了车辆的侧向受力，并且将约束条件转换为加速度圆（Acceleration Circle Constraint）。应用该模型能够顺利完成避障任务。点质量模型相比于普通自行车模型计算量更小，可以有效地减少车辆非线性动力学模型的复杂性。将车辆动力学模型近似为点质量模型，同时轮胎摩擦圆约束也被映射到加速度约束上，以及其他的一些技巧均可用于简化计算。

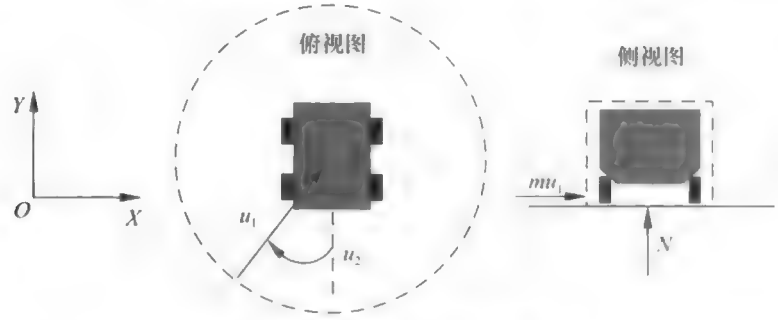


图 1.6　点质量模型受力分析

（3）运动规划算法及存在的问题小结

从上面可以看出，轨迹规划方面的研究一般都按照局部路径规划方法进行，然后在考虑多种约束条件的情况下对其进行轨迹化。与低速行驶的移动机器人不同的是，无人驾驶车辆的路径和轨迹规划更多采用多项式行驶曲线的方式进行拟合，同时考虑了执行机构的约束。这种规划的特点是能够保证车辆平顺行驶，不会出现突然转向。为了进一步提高系统的实时性，微分平坦、点质量模型等相关理论也被应用在了轨迹规划的过程中。

普通规划方法在高速无人车辆运动规划中普遍存在的问题是实时性达不到要求，特别是紧急情形下的危险规避运动规划对实时性要求更高。基于动力学优化模型的规划方法均考虑了轮胎摩擦圆约束，采用受加速度约束的点质量模型来近似轮胎摩擦圆约束，但没有对高速车辆动力学模型结合运动规划做深入的最优分析。侧倾约束下的运动规划方法研究较少，且主要集中在越野环境，仅仅是根据阈值窗口范围进行运动规划，也同样没有根据车辆动力学理论的稳定性最优原则产生运动轨迹。此外，同时考虑滑移与侧倾的研究则更少。

点质量模型最优控制特性分析也是运动规划中的一个重要因素。用加速度值有界且方向可控的点质量模型进行最优控制特性分析和证明，能保证建立在此基础上的相关算法的可靠应用。Bryson 和 Ho 给出了几种情形的证明并且将其应用到运载火箭的发射中，用于推力控制分析[44]，可以供地面车辆的运动规划最优特性分析参考。

1.3.2　轨迹跟踪控制的模型约束

轨迹跟踪控制的目的是让无人驾驶车辆跟踪由规划算法得到的轨迹。其主要任务是根据车辆的运动学和动力学约束输出相应的控制参数，如前轮偏角、车轮制动力等。目前应用较多的控制算法有PID控制算法、滑模控制算法、神经网络控制算法等。这些方法对参数和环境的依赖程度较高，当环境发生较大变化时不能很好地适应新状态下的轨迹跟踪；同时，地面无人车辆在运动过程中受到运动学约束以及执行机构的约束，在高速运动情况下还需要考虑相应的动力学约束。上述方法对这些约束难以处理。由于基于模型预测控制的轨迹跟踪算法对未来轨迹的预测和处理多目标约束条件的能力较强，故逐渐成为研究的热点。

在应用车辆模型进行预测时，是否考虑轮胎滑移是一个非常重要的约束条件。这也是实际车辆测试的一个难点问题。轮胎滑移率也是驱动防滑系统（ASR）及防抱死制动系统（ABS）的关键参数。考虑到轮胎柔度和摩擦力饱和值，轮胎纵横向耦合摩擦力（驱动力）模型一般为滑移率的非线性函数。目前应用较广的是Pacejka提出的魔术公式轮胎模型。

（1）非完整动力学约束轨迹跟踪

非完整（Nonholonomic）动力学约束就是假设车辆在运动过程中没有滑移。很多在平坦路面运动的无人车辆或移动机器人都用这种假设设计轨迹跟踪控制器，特别是纯跟踪算法（Pure Pursuit），已经被成功地应用在很多机器人轨迹跟踪控制中，包括获得2005年DAPAR沙漠挑战赛冠军的斯坦福大学的Stanley赛车[45]。

（2）摩擦力与滑移率非线性约束轨迹跟踪

带有纵横向耦合滑移的非线性轮胎模型在轨迹跟踪控制器中一般利用数值优化计算方法进行解算。文献[46]应用凸二次锥规划优化数值计算方法，而且在控制中使用了非线性耦合模型的简化模型。文献[47]用预测控制方法为自主车辆设计适应轮胎摩擦极限的轨迹跟踪控制器，应用了耦合非线性轮胎模型，该模型较复杂。Velenis和Frazzoli通过推导出无人驾驶车辆在转弯时的速度、轮胎侧偏角、横摆角速度等参数，在仿真实验中应用滑模控制方法能很好地控制车辆在漂移状态下行驶[48]。其主要不足是只能稳定地控制均匀漂移的运动并且还无法跟踪时变曲率轨迹。

（3）车辆动力学模型简化对模型预测控制的影响

预测模型的选择也是基于模型预测控制的轨迹跟踪算法需要考虑的关键问题之一。使用复杂的模型能够更好地对车辆输出进行预测，但同时也增大了控

制器的计算量，实时性难以保证；而使用简单的模型有可能导致跟踪失败。针对这个问题，文献［49］分别基于2自由度自行车模型和14自由度动力学模型设计了轨迹跟踪控制器，然后对轨迹跟踪效果进行对比，结果如图1.7所示。其中，图1.7（a）为没有考虑安全约束，而图1.7（b）中两者都加入安全约束。可以看出，基于14自由度模型的控制器能够具备更好的避障效果；但加入了14自由度模型中的安全约束之后，2自由度模型也能具备相近的效果并且计算时间优势明显。因此，复杂的预测模型并不是最好的选择，对车辆动力学模型进行合理简化并且选择满足行驶工况的约束条件才是研究的重点。也就是说，在模型预测控制中，模型约束条件能够表达控制对象的真实物理限制即可。

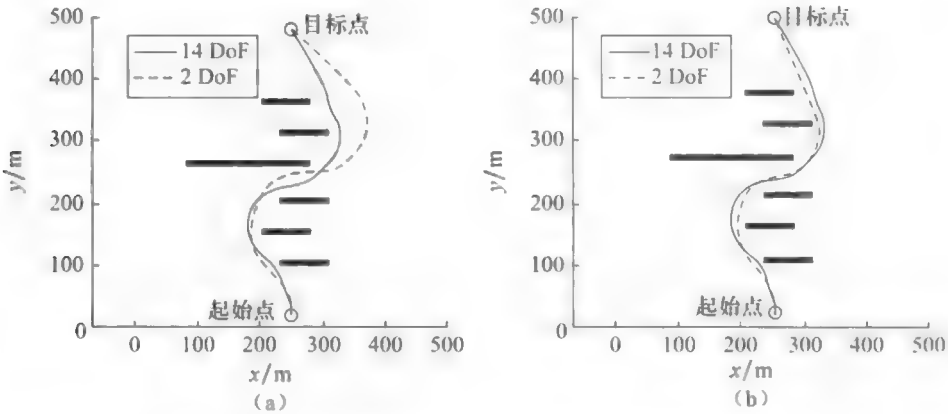


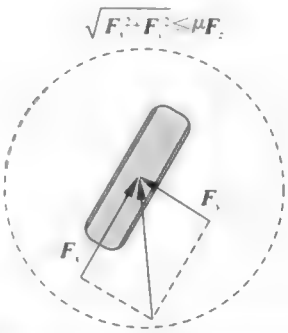
图1.7　2自由度模型与14自由度模型跟踪结果对比

（a）未加入安全约束；（b）加入安全约束

上面关于模型复杂度对控制效果的分析为控制过程中的模型简化提供了依据。文献［50］通过对轮式移动机器人的运动学模型在目标轨迹点附近进行线性化，得到了离散的系统状态空间模型；采用线性模型预测控制算法进行轨迹跟踪，并通过仿真和实车实验分别验证了算法的可靠性。

在动力学建模方面，麻省理工学院（Massachusetts Institute of Technology，MIT）、斯坦福大学（Stanford University）以及韩国釜山国立大学等做了深入的研究。MIT的Robotic Mobility Group课题组通过微分平坦理论、点质量模型对车辆动力学模型进行合理简化，应用轮胎的摩擦圆约束（如图1.8所示），提高了系统的实时性，实现在高速情况下的地面无人车辆危险规避[51,52]。轮胎摩擦圆代表轮胎与地面摩擦所能产生的最大牵引力。由于牵引力可以为任意方向，以轮胎与地面接触中心点为圆心，最大牵引力为半径得到的圆，就是

轮胎摩擦圆　摩擦圆约束是指轮胎与地面摩擦时纵向与横向产生的耦合力必须位于摩擦圆内。车辆转弯、加速、减速时，如果耦合力超出了摩擦圆，车辆就会产生滑移。轮胎摩擦圆会因为路面的状况、轮胎的接地负荷或者轮胎的性能而改变。例如在冰雪或湿滑的路面，轮胎摩擦圆比在常规路面行驶时小。



μ 轮胎与地面具有饱和特征的摩擦系数　F_z 轮胎横向摩擦力
F_x 轮胎纵向摩擦力　F_z 轮胎法向作用力

图 1.8　轮胎摩擦圆约束

在如图 1.9 所示的高速车辆换道避障运动轨迹规划中，D_2 是车辆与前方障碍的纵向距离，D_1 是本车道与旁车道的距离，那么危险规避运动规划问题可表述为：首先计算在满足各轮胎摩擦圆约束和障碍约束的条件下，车辆能够安全换到距离为 D_1 的车道的最小纵向距离 $D_{2\min}$。当调整到 $D_2 > D_{2\min}$ 时，计算车辆换道运动轨迹。文献[53]利用二次锥规划方法来确定作用在车辆上的最优力和力矩。车辆用刚体表示，并用由两步优化方法得到的转向和制动命令来控制高阶车辆模型跟踪期望轨迹。

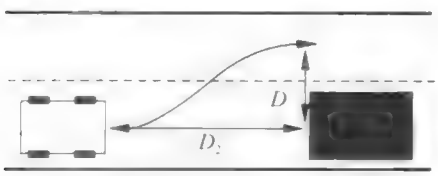


图 1.9　前方有障碍（车辆或其他障碍）的换道行驶情形

Stanford 的 Craig 等在二自由度自行车模型基础上提出了仿射力矩输入模型（Affine Force Input），并将横摆角速度和轮胎侧偏角约束以包线的形式纳入控制器（如图 1.10 所示），使车辆能够在极限情况下完成轨迹跟踪，实现车辆

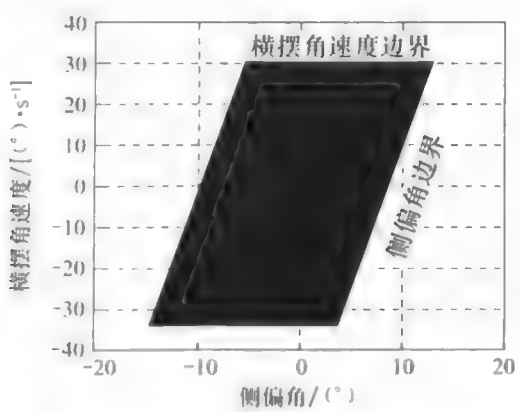


图 1.10　侧偏角和横摆角速度包线约束

性能的最大化利用[54]。文献［55］则在动力学建模时考虑了车轮转角函数随速度的变化，提高了车辆在高速行驶情况下的控制精度。

（4）微分平坦方法在非线性运动控制模型简化中的应用

微分平坦在无人（空中或地面）车辆或者机器人轨迹跟踪控制中得到了一定的关注。微分平坦是一个优化控制问题。从理论上分析，它与全状态反馈线性优化是等价的。微分平坦系统的定义：如果一个系统能找到平坦输出，使系统的状态和输入都可以用平坦输出及其有限阶导数表示，则该系统被称为平坦系统[30,56]。平坦系统的一个好处就是能将非线性系统的微分约束映射为平滑几何约束，有效减少优化空间的维数。对于车辆非线性系统，利用微分平坦输出，能将同时满足运动学约束（速度）和动力学约束（加速度）的 Kinodynamic 规划与控制问题简化为计算光滑轨迹的几何规划与控制问题。

另外，微分平坦简化模型轨迹跟踪的控制稳定性非常重要。这是滚动优化评价的基础。文献［51］完成了固定加速度横摆动力学模型在跟踪固定曲率轨迹的控制稳定性分析，但没有涉及时变加速度模型及时变曲率轨迹。

（5）侧倾约束对轨迹跟踪控制的影响

目前已经有商用化的车辆（特别是挂车及高档越野轿车）防侧倾控制系统。其原理主要是综合前轮偏角与差动制动方法输入控制量、主动悬架控制技术调整重心位置、发动机扭矩控制与 ABS 系统结合使用等方法达到防侧倾的目的。对于在越野环境中运行的车辆或者高速车辆紧急避险转向时侧倾与滑移约束相结合的轨迹跟踪控制稳定性分析是非常重要的。Iagnemma[57] 采用能量成形控制（Energy Shaping Control）分析了近似成倒立摆的车辆模型侧倾控制稳定性，借鉴了倒立摆的控制稳定性分析（如图 1.11 所示），虽然忽略了横摆滑移非线性，但能量成形控制稳定性方法体现了较好的易用性。

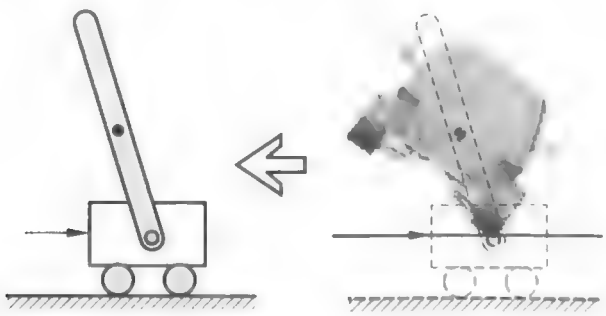


图 1.11　轨迹跟踪时车辆侧倾约束倒立摆模型

（6）模型预测控制算法实时性

与移动机器人不同，无人车辆处于高速运动状态，车辆不可能停下来或者减速等待规划或控制器的输出结果。基于模型预测控制的轨迹跟踪控制器实时性问题一方面受到所采用动力学模型的影响，另外一方面则是自身计算方式的影响。目前模型预测控制的基本思想都是在线求解序列二次规划问题，而当系统较为复杂或者约束量比较多时，计算速度不能满足实时控制的需求。提高计算速度的一类方法是将二次规划问题完全用控制量表示，并假定在某一段控制时域内控制量不变。这样可以大大减少优化过程中的变量，从而提高计算速度[57]。另外一种重要的方法被称为“暖启动”[59]，即在每一步开始计算的时候根据上一步的预测结果进行初始化。如果运用得当的话，这种方法将会使总计算次数减少为原来的1/5。

随着模型预测控制算法在各领域的应用愈加广泛，有学者将前期的一些研究成果进行集成，形成了方便研究者使用的工具箱。Fast MPC是目前应用较为普遍的快速求解工具箱。它的基本思想是在求解器得到符合目标精度的结果后就提前结束迭代过程，即不再要求每一步计算过程都得到最优解。实验结果表明，应用该工具箱处理含有450个变量和1 284个约束条件的复杂系统时，可以达到200Hz的计算速度[60]。多参数规划工具箱（Multi-Parametric Toolbox，MPT）是一类快速求解线性规划或者二次规划问题的工具箱。其基本原理是通过应用多参数规划方法得到约束最优控制序列与状态之间的显式函数关系，减少规划问题求解的工作量[61,62]。

（7）轨迹跟踪算法及存在的问题小结

基于模型预测控制的轨迹跟踪算法已经有了一定的研究，对车辆动力学的建模已经比较成熟，仿真和实验都验证了算法的可行性与可靠性。主要突破点为：

1. 对动力学模型合理简化，减少控制器的在线计算量，提高控制系统的实时性。

2. 对高速情况下的动力学模型进行分析，提高控制系统在车辆高速行驶时的控制精度。

高速车辆在危险障碍规避时的运动轨迹跟踪控制过程中，为保证控制算法满足非线性约束和实时性要求，采用了车辆动力学模型的降维方法；但目前设计的运动轨迹跟踪控制器中大都采用数值优化解算非线性轮胎模型，模型的复杂性增加了计算负担。

非线性系统的微分平坦输出特性能有效地将运动轨迹的动力学和运动学Kinodynamic约束映射为几何平滑约束，简化了非线性系统规划与控制计算复

杂度，同时将车辆前轮附近的振动中心位置属性作为非线性自行车模型的平坦输出，在高速滑动漂移的运动轨迹跟踪控制中有很好的研究价值。

时变加速度车辆动力学模型（横摆动力学及其他约束等效模型）跟踪时变曲率轨迹的控制稳定性分析尚待突破。

1.4 本书内容与结构说明

虽然前面列出了地面无人驾驶车辆运动规划与控制中的一系列挑战性问题，但我们在研究过程中发现模型预测控制方法能够充分考虑这些挑战性问题中涉及的车辆模型约束问题，并进行了一些初步的探索[16,17,23,81,82,83]。本书对这些初步的研究成果进行了系统的总结，一方面作为团队进一步深入研究的基础材料，另一方面也可以作为模型预测控制理论应用研究的参考资料。另外，鉴于模型预测控制理论的数学抽象性较强、入门和理解需要较长时间，为了方便读者充分理解本书内容，书中给出了所有算法 Matlab 代码，同时仿真实例也都列出了详细步骤。

本书假定无人驾驶车辆的感知系统能够对行驶环境进行充分的理解，能够得到障碍物、道路条件等环境信息，同时具备全局路径规划能力；车辆平台具备自动驾驶控制能力，横向与纵向控制系统能够控制车辆转向与速度，并且能向控制系统反馈车辆外部和内部状态信息。

第 2 ~ 7 章的内容如下：

第 2 章介绍用于地面无人驾驶车辆运动规划与控制的车辆运动学模型和动力学模型。车辆类型选择前轮转向的乘用汽车，介绍了车辆运动学建模过程并对模型进行了验证；介绍了侧向动力学模型和轮胎模型，并给出了 Matlab 代码实例验证。

第 3 章首先阐述模型预测控制基本概念，扩展到两类常用的模型预测控制方法，并以工程实例进一步加深对这种方法的认识。然后分别对线性模型预测控制和非线性模型预测控制进行介绍，并对仿真源代码进行分析，让读者能够尽快熟悉并编写自己的控制算法。

第 4 章从运动学模型角度出发，开发具有轨迹跟踪功能的控制器，在给定期望轨迹点的情况下，完成对期望轨迹的跟踪控制，并详细说明搭建 Simulink/CarSim 联合仿真平台的过程，对所设计的控制器进行仿真验证。本章将理论与实践相结合，在介绍完轨迹跟踪控制器的理论算法后，通过实例进一步加深对理论的认知。从 CarSim 软件使用和 Matlab 代码分析的角度，让读者对轨迹跟踪控制的设计与使用有深入的理解。

第5章以无人驾驶车辆的主动转向控制为应用背景，建立以车辆动力学模型为基础的模型预测控制器，结合车辆高速运动下的各类约束，进一步讨论复杂约束条件下模型预测控制的求解方法，并在Simulink/CarSim联合仿真平台上验证算法的有效性。

第6章介绍地面无人驾驶车辆结合路径规划层的轨迹跟踪控制器，车辆类型选择前轮转向的乘用汽车。首先，介绍了基于车辆点质量模型的MPC轨迹规划控制器，规划出满足车辆动力学约束并实现避障的可行轨迹，并给出了相应的S函数。其次，介绍了在车辆动力学模型的基础上的MPC路径跟踪控制器，并给出了S函数代码。再次，详细介绍了运用Simulink/CarSim进行联合仿真的实例验证。

第7章是基于模型预测控制思想设计的实际车辆控制中运用的单步航向预估控制方法，虽然没有进行优化，但在考虑模型约束的基础上，结合PID控制方法，在单片机程序中实现了无人驾驶车辆航向单步预测，有较好的控制效果。

第2章

车辆运动学与动力学建模

无人驾驶车辆运动规划与控制需要通过对车辆运动学或者动力学系统的控制来实现。从第1章的介绍可以看出，如果规划阶段能够考虑车辆运动学和动力学约束，那么运动跟踪控制性能会更好。建立合理的车辆系统模型不仅是设计模型预测控制器的前提，也是实现车辆道路跟踪功能的基础。因此，在建立模型预测控制器时，必须根据无人驾驶车辆的具体行驶工况，通过选取合适的控制变量，建立能够准确描述无人驾驶车辆运动关系约束的运动学模型和描述动力学约束的动力学模型。

车辆在地面运动的动力学过程是非常复杂的，为了尽量准确描述车辆运动，需要建立复杂的微分方程组，并用多个状态变量来描述其运动。在第1章已经讨论到，用于模型预测控制的模型只要能够表现出车辆运动学与动力学约束，就可以使模型预测控制器实现预定控制目的。特别是在规划阶段，为了保证规划算法的实时性，约束简化和近似就是一种非常重要的手段，比如轮胎摩擦圆约束和点质量模型。因此，过于复杂的模型并不是研究的重点。本章从无人驾驶车辆路径重规划和道路跟踪控制的角度对车辆系统进行建模，建立能够尽量准确反映车辆运动特性，并且有利于模型预测控制器设计的简化车辆运动学模型和动力学模型[4]。在此基础上，对所建立的模型进行验证。

2.1 车辆运动学建模及验证

运动学是从几何学的角度研究物体的运动规律，包括物体在空间的位置、

速度等随时间而产生的变化。在车辆路径规划算法中应用运动学模型，可以使规划出的路径切实可行、并满足行驶过程中的运动学几何约束；同时，在良好路面的低速行驶工况下，一般不需要考虑车辆稳定性控制等动力学问题。基于运动学模型设计的路径跟踪控制器具备可靠的控制性能。

2.1.1 车辆运动学建模

车辆转向运动模型如图 2.1 所示。在惯性坐标系 OXY 下，(X_r, Y_r) 和 (X_f, Y_f) 分别为车辆后轴和前轴轴心的坐标，φ 为车体的横摆角（航向角），δ_f 为前轮偏角，v_r 为车辆后轴中心速度，v_f 为车辆前轴中心速度，l 为轴距（注意，变量下标 f 代表 front——前，r 代表 rear——后，下同）。

图 2.2 所示为车辆转向过程示意图，R 为后轮转向半径，P 为车辆的瞬时转动中心，M 为车辆后轴轴心，N 为前轴轴心。此处假设转向过程中车辆质心侧偏角保持不变，即车辆瞬时转向半径与道路曲率半径相同。

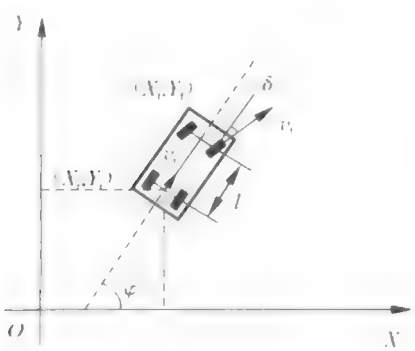


图 2.1 车辆运动模型

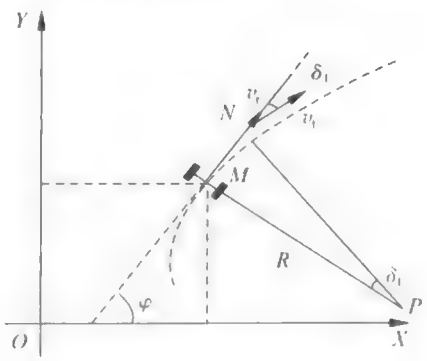


图 2.2 车辆前轮转向示意图

在后轴行驶轴心 (X_r, Y_r) 处，速度为：

$$v_r = \dot{X}_r\cos\varphi + \dot{Y}_r\sin\varphi \tag{2.1}$$

前、后轴的运动学约束为：

$$\begin{cases} \dot{X}_f\sin(\varphi + \delta_f) - \dot{Y}_f\cos(\varphi + \delta_f) = 0 \\ \dot{X}_r\sin\varphi - \dot{Y}_r\cos\varphi = 0 \end{cases} \tag{2.2}$$

由式（2.1）和式（2.2）联合可得：

$$\begin{cases} \dot{X}_r = v_r\cos\varphi \\ \dot{Y}_r = v_r\sin\varphi \end{cases} \tag{2.3}$$

根据前后轮的几何关系可得：

$$\begin{cases} X_f = X_r + l\cos\varphi \\ Y_f = Y_r + l\sin\varphi \end{cases} \tag{2.4}$$

将式（2.3）和式（2.4）代入式（2.2），可解得横摆角速度为：

$$\omega = \frac{v_r}{l}\tan\delta_f \tag{2.5}$$

式中，ω 为车辆横摆角速度；同时，由 ω 和车速 v_r 可得到转向半径 R 和前轮偏角 δ_f：

$$\begin{cases} R = v_r/\omega \\ \delta_f = \arctan(l/R) \end{cases} \tag{2.6}$$

由式（2.3）和式（2.5）可得到车辆运动学模型为：

$$\begin{bmatrix} \dot{X}_r \\ \dot{Y}_r \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos\varphi \\ \sin\varphi \\ \tan\delta_f/l \end{bmatrix} v_r \tag{2.7}$$

该模型可被进一步表示为更为一般的形式：

$$\dot{\boldsymbol{\xi}}_{kin} = f_{kin}(\boldsymbol{\xi}_{kin}, \boldsymbol{u}_{kin}) \tag{2.8}$$

其中，状态量 $\boldsymbol{\xi}_{kin} = [X_r, Y_r, \varphi]^T$，控制量 $\boldsymbol{u}_{kin} = [v_r, \delta_f]^T$。在无人驾驶车辆的路径跟踪控制过程中，往往希望以 $[v_r, \omega]$ 作为控制量，将式（2.5）代入式（2.7）中，该车辆运动学模型可以被转换为如下形式：

$$\begin{bmatrix} \dot{X}_r \\ \dot{Y}_r \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos\varphi \\ \sin\varphi \\ 0 \end{bmatrix} v_r + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega \tag{2.9}$$

2.1.2 车辆运动学模型验证

为了验证所建立的车辆运动学模型，在 Matlab/Simulink 环境中搭建该运动学模型，在相同的输入条件下与 CarSim 中所建立的整车模型进行对比分析。其中，相同的输入条件是指前轮偏角与车速随时间的变化历程相同，输出均为车辆位置和航向。

车辆基本参数设置为：

轴距：$l = 2.7$ m；

初始状态：$\boldsymbol{\xi}_{kin} = [0, 0, 0]$。

用于模型验证的输入信号随时间的变化历程如图 2.3 所示，其中图 2.3（a）

为车速随时间的变化历程，图 2.3（b）为前轮偏角随时间的变化历程。上述所建立的运动学模型和 CarSim 整车模型的输出信号如图 2.4 所示。其中图 2.4（a）为两者在给定的输入条件下输出的位置信息对比，图 2.4（b）为两者的航向角信息对比。如果对 CarSim 和 Simulink 的联合使用方法不熟悉，则可在看完第 4 章的详细仿真步骤说明后，再完成模型的验证。

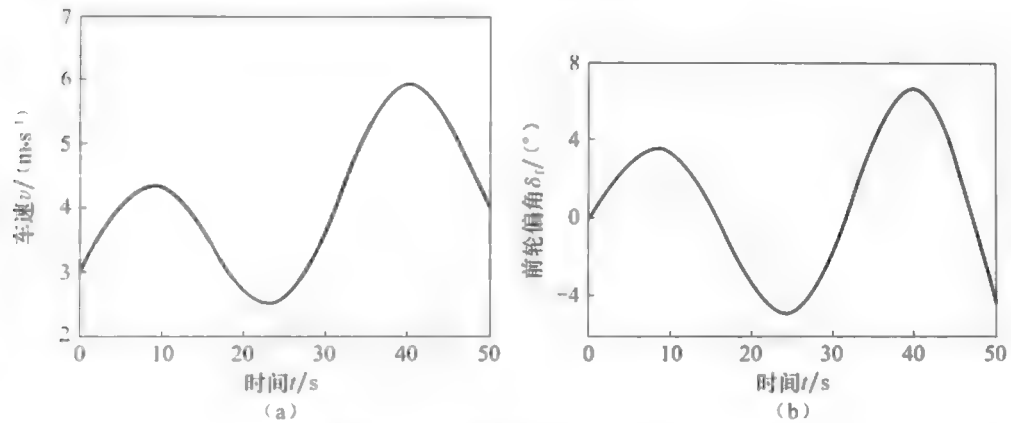


图 2.3 模型验证的输入信号

（a）车速随时间的变化历程；（b）前轮偏角随时间的变化历程

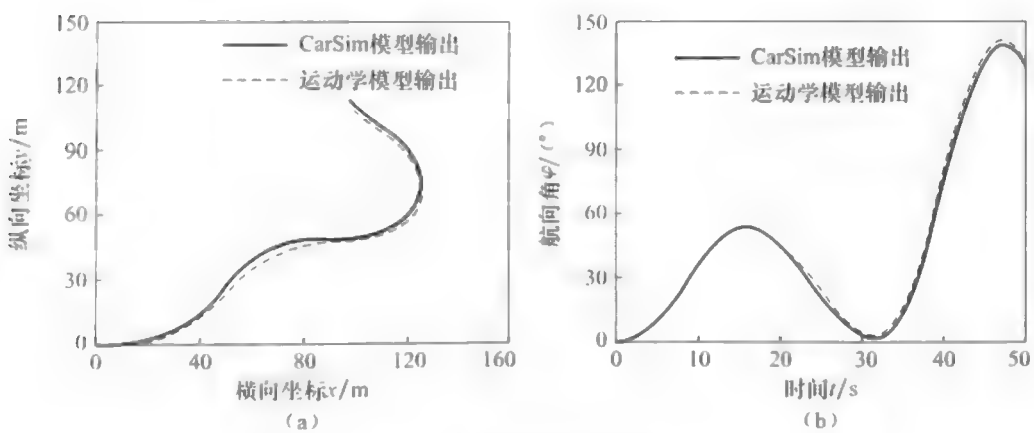


图 2.4 模型输出结果对比

（a）车辆位置对比；（b）车辆航向角对比

从图 2.4 中的结果对比可以看出，在相同的速度和前轮偏角输入下，运动学模型的车辆位置和航向角与 CarSim 输出的结果非常吻合。也就是说，式（2.8）表示的模型能够较好地反映车辆行驶时的运动学特性。在后续规划

算法以及低速轨迹跟踪控制算法中，将在仿真实验中使用该模型。

2.2 车辆动力学建模及验证

车辆整车动力学模型一般包括用于分析车辆平顺性的质量－弹簧－阻尼模型和分析车辆操纵稳定性的车辆－轮胎模型，两者研究的侧重点不同，平顺性分析的重点是车辆的悬架特性，而车辆操纵稳定性分析的重点是车辆纵向及侧向动力学特性。本书主要研究目标是使车辆快速而稳定地跟踪期望路径，属于车辆操纵稳定性问题，因此对于悬架特性不做深入探究；同时，本书所建立的动力学模型主要是作为模型预测控制器中的预测模型使用，需要在较为准确地描述车辆动力学过程的基础上尽可能进行简化，以减少控制算法的计算量。

综合上述分析，在进行车辆动力学建模时，进行以下理想化的假设：

① 假设无人驾驶车辆在平坦路面上行驶，忽略车辆垂向运动。

② 悬架系统及车辆是刚性的，忽略悬架运动及其对耦合关系的影响。

③ 只考虑纯侧偏轮胎特性，忽略轮胎力的纵横向耦合关系。

④ 用单轨模型来描述车辆运动，不考虑载荷的左右转移。

⑤ 假设车辆行驶速度变化缓慢，忽略前后轴的载荷转移。

⑥ 忽略纵向和横向空气动力学。

2.2.1 车辆单轨模型

基于以上6点理想假设，平面运动车辆只具有3个方向的运动，即纵向、横向和横摆运动。设定车辆为前轮驱动。满足以上设定的平面运动车辆单轨模型如图2.5所示。其中，坐标系 $oxyz$ 为固定于车身的车辆坐标系，xoz 处于车辆左右对称的平面内，车辆质心所在点为坐标原点 o，x 轴为沿车辆纵轴，y 轴与车辆纵轴方向垂直，而 z 轴满足右手法则，垂直于 xoy 且向上。坐标系 OXY 为固定于地面的惯性坐标系，也满足右手法则。图2.5中关于轮胎受力定义如下：

$\boldsymbol{F}_{lf}$，$\boldsymbol{F}_{lr}$：前、后轮胎受到的纵向力。

$\boldsymbol{F}_{cf}$，$\boldsymbol{F}_{cr}$：前、后轮胎受到的侧向力。

$\boldsymbol{F}_{xf}$，$\boldsymbol{F}_{xr}$：前、后轮胎受到的 x 方向的力。

$\boldsymbol{F}_{yf}$，$\boldsymbol{F}_{yr}$：前、后轮胎受到的 y 方向的力。

根据牛顿第二定律，分别得沿 x 轴、y 轴和绕 z 轴的受力平衡方程

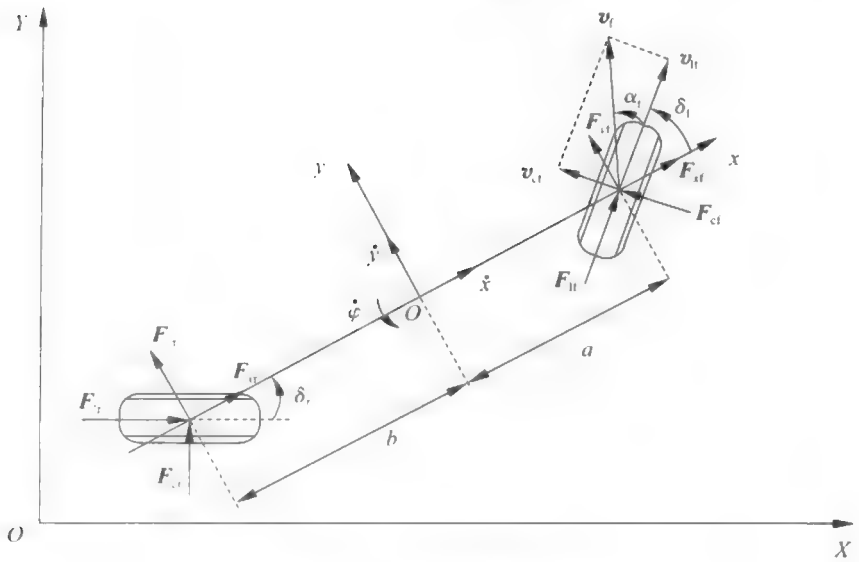


图 2.5 车辆单轨模型

在 x 轴方向上：

$$m\ddot{x} = m\dot{y}\dot{\varphi} + 2F_{xf} + 2F_{xr} \tag{2.10}$$

在 y 轴方向上：

$$m\ddot{y} = -m\dot{x}\dot{\varphi} + 2F_{yf} + 2F_{yr} \tag{2.11}$$

绕 z 轴方向上：

$$I_z\ddot{\varphi} = 2aF_{yf} - 2bF_{yr} \tag{2.12}$$

式中，a、b 分别为质心到前、后轴的距离，m 为车辆整备质量，I_z 为车辆绕 z 轴的转动惯量

轮胎在 x 方向和 y 方向上受到的合力与纵、侧向力的转换关系如下：

$$F_{xf} = F_{lf}\cos\delta_f - F_{cf}\sin\delta_f \tag{2.13}$$

$$F_{xr} = F_{lr}\cos\delta_r - F_{cr}\sin\delta_r \tag{2.14}$$

$$F_{yf} = F_{lf}\sin\delta_f + F_{cf}\cos\delta_f \tag{2.15}$$

$$F_{yr} = F_{lr}\sin\delta_r + F_{cr}\cos\delta_r \tag{2.16}$$

轮胎的纵向力、侧向力可以表示为轮胎侧偏角、滑移率、路面摩擦系数和垂向载荷等参数的复杂函数：

$$F_l = f_l(\alpha, s, \mu, F_z) \tag{2.17}$$

$$F_c = f_c(\alpha, s, \mu, F_z) \tag{2.18}$$

式中，α 为轮胎侧偏角，s 为滑移率，μ 为路面摩擦系数，F_z 为轮胎所受到的垂向载荷。

轮胎的侧偏角 α 可以由几何关系计算得到：

$$\alpha = \tan^{-1}\frac{v_c}{v_l} \tag{2.19}$$

式中，v_c 和 v_l 为轮胎在侧向、纵向的速度，可以用坐标系方向的速度 v_x 和 v_y 表示：

$$v_l = v_y\sin\delta + v_x\cos\delta \tag{2.20}$$

$$v_c = v_y\cos\delta - v_x\sin\delta \tag{2.21}$$

式中，δ 为轮胎偏转角。

轮胎的速度往往难以直接获取，一般可以通过车辆速度计算得到。根据图 2.5 中的速度关系可以推导出以下转换关系：

$$v_{yf} = \dot{y} + a\dot{\varphi} \quad v_{yr} = \dot{y} - b\dot{\varphi} \tag{2.22}$$

$$v_{xf} = \dot{x} \quad v_{xr} = \dot{x} \tag{2.23}$$

轮胎在地面上的滑移率 s 可以由以下算式计算：

$$s = \begin{cases} \dfrac{r\omega_t}{v} - 1 (v > r\omega_t, v \neq 0) \\ 1 - \dfrac{v}{r\omega_t} (v < r\omega_t, \omega_t \neq 0) \end{cases} \tag{2.24}$$

式中，r 为车轮半径，ω_t 为车轮旋转角速度。

假设车辆行驶速度变化缓慢，忽略前后轴的载荷转移，可以通过以下算式计算得到车辆前、后轮胎所受到的垂向载荷：

$$F_{zf} = \frac{bmg}{2(a+b)} \tag{2.25}$$

$$F_{zr} = \frac{amg}{2(a+b)} \tag{2.26}$$

最后，考虑车身坐标系与惯性坐标系之间的转换关系，可得：

$$\dot{Y} = \dot{x}\sin\varphi + \dot{y}\cos\varphi \tag{2.27}$$

$$\dot{X} = \dot{x}\cos\varphi - \dot{y}\sin\varphi \tag{2.28}$$

结合式（2.10）~式(2.28)，可以得到车辆非线性动力学模型。通过算式间的代换，除了路面摩擦系数 μ 和滑移率 s 外，其他参数都可以由车辆状态信息计算得到。路面摩擦系数 μ 为道路固有信息，给定道路条件后就能获取。把滑移率 s 作为系统的控制量，将有效改善车辆在低附着路面的行驶性能，但对于滑移率的控制本身就是一个复杂的控制问题。因此，假设被控车辆具备良好的防抱死制动系统（ABS），滑移率始终保持在最佳工作点，将系统描述为以

下状态空间表达式：

$$\begin{aligned}\dot{\boldsymbol{\xi}}_{\mathrm{dyn}} &= f_{\mathrm{dyn}}(\boldsymbol{\xi}_{\mathrm{dyn}}, \boldsymbol{u}_{\mathrm{dyn}}) \\ \boldsymbol{\eta}_{\mathrm{dyn}} &= h_{\mathrm{dyn}}(\boldsymbol{\xi}_{\mathrm{dyn}})\end{aligned} \tag{2.29}$$

在该系统中，状态量选取为 $\boldsymbol{\xi}_{\mathrm{dyn}} = [\dot{y},\ \dot{x},\ \varphi,\ \dot{\varphi},\ Y,\ X]^{\mathrm{T}}$，控制量选取为 $\boldsymbol{u}_{\mathrm{dyn}} = \delta_{\mathrm{f}}$（仅考虑前轮转向车辆，$\delta_{\mathrm{r}}$ 视为0），输出量选取为 $\boldsymbol{\eta}_{\mathrm{dyn}} = [\varphi,\ Y]^{\mathrm{T}}$。在实际的控制过程中，路面摩擦系数 μ 和滑移率 s 视为已知量，该模型即模型预测控制器中预测模型的基础。

2.2.2 轮胎模型

在进行车辆动力学仿真时，轮胎所受的垂直力、纵向力、侧向力和回正力矩对汽车的平顺性、操纵稳定性和安全性起着重要作用。由于轮胎结构复杂，动力学性能呈非线性，选择符合实际又便于使用的轮胎模型是建立车辆模型并进行动力学仿真的关键。主要的轮胎模型可以分为理论轮胎模型、经验轮胎模型和物理轮胎模型等。其中，Pacejka 提出的以魔术公式（Magic Formula，MF）为基础的经验轮胎模型，运用三角函数的组合公式拟合轮胎试验数据，描述轮胎的纵向力 $\boldsymbol{F}_{\mathrm{t}}$，侧向力 $\boldsymbol{F}_{\mathrm{c}}$，回正力矩 $\boldsymbol{M}_{\mathrm{z}}$，翻转力矩 $\boldsymbol{M}_{\mathrm{x}}$，阻力矩 $\boldsymbol{M}_{\mathrm{r}}$ 与侧偏角 α、滑移率 s 之间的定量关系，以及纵向力、侧向力的联合作用工况，能够表达不同驱动情况时的轮胎特性[65]。魔术公式的一般表达式为：

$$Y(x) = D\sin\{C\arctan[Bx - E(Bx - \arctan(Bx))]\} \tag{2.30}$$

式中，系数 B、C、D 依次由轮胎的垂直载荷和外倾角确定；Y 为输出变量，可以是纵向力 $\boldsymbol{F}_{\mathrm{t}}$ 或侧向力 $\boldsymbol{F}_{\mathrm{c}}$ 或者回正力矩；x 为输入变量，在不同的情况下分别表示轮胎的侧偏角 α 或纵向滑移率 s；B 为刚度因子；C 为形状因子；D 为峰值因子；E 为曲率因子。

在实际应用中，由于帘布层转向效应、侧偏力作用或滚动阻力会引起偏移，通常还会引入垂直偏移 S_{v} 和水平偏移 S_{h}（见图2.6）。

Pacejka 轮胎模型认为，轮胎在垂直、侧面方向上是线性的，阻尼为常量，这在侧向加速度 $a_{\mathrm{c}} \leqslant 0.4g$、轮胎侧偏角 $\alpha \leqslant 5°$ 的情景下对常规轮胎具有很高的拟合精度。此外，魔术公式具有较好的健壮性，在极限值以外一定程度内仍可使用，可以对有限工况进行外推且具有较好的置信度。

利用魔术公式计算轮胎的纵向力及侧向力的关系式分别介绍如下：

（1）轮胎纵向力计算方法

$$\boldsymbol{F}_{\mathrm{t}} = D\sin[C\arctan(Bx_{\mathrm{t}} - E(Bx_{\mathrm{t}} - \arctan(Bx_{\mathrm{t}})))] + S_{\mathrm{v}} \tag{2.31}$$

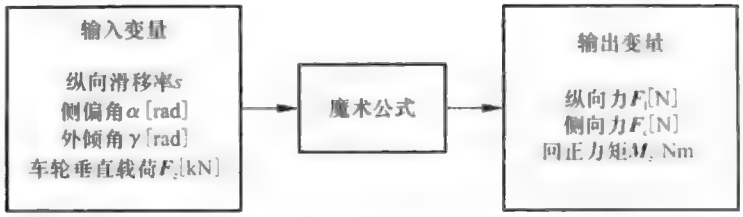


图 2.6　基于魔术公式的轮胎模型的输入和输出变量

式中，$x = s + S_h$，s 为纵向滑移率（负值出现在制动态，-100 表示车轮抱死），S_h 为曲线的水平方向漂移，S_v 为曲线的垂直方向漂移

C 为曲线的形状因子，$C = B_0$。

D 为曲线巅因子，表示曲线的最大值，$D = B_1F_z^2 + B_2F_z$，其中 F_z 为轮胎受到的垂向载荷。

B 为刚度因子，$B = (B_3F_z^2 + B_4F_z) \times e^{-B_5F_z}/(C \times D)$

E 为曲线的曲率因子，表示曲线最大值附近的形状，$E = B_6F_z^2 + B_7F_z + B_8$

S_h 为曲线的水平方向漂移：$S_h = B_9F_z + B_{10}$

S_v 为曲线的垂直方向漂移：$S_v = 0$。

在纯滑移率作用下轮胎的纵向力如图 2.7 所示

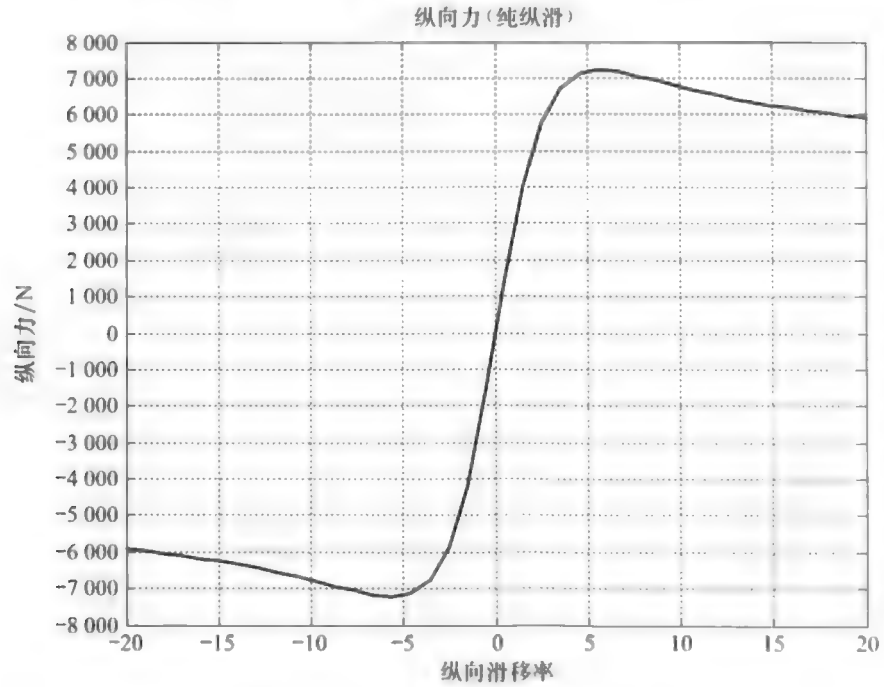


图 2.7　Pacejka'89 轮胎纵向力示例

（2）轮胎侧向力计算

$$F_c = D\sin[C\arctan(Bx - E(Bx - \arctan(Bx)))] + S_v \tag{2.32}$$

式中，$x = \alpha + S_h$，α 为轮胎侧偏角，S_h 为曲线的水平方向漂移，S_v 为曲线的垂直方向漂移。

C 为曲线的形状因子，$C = A_0$。

D 为曲线峰因子，表示曲线的最大值，$D = A_1F_z^2 + A_2F_z$。

B 为刚度因子，$B = A_3\sin\left(2\arctan\dfrac{F_z}{A_4}\right) \times (1 - A_5 \mid \gamma \mid)/(C \times D)$，其中 γ 为轮胎外倾角。

E 为曲线的曲率因子，$E = A_6F_z + A_7$。

S_h 为曲线的水平方向漂移：$S_h = A_9F_z + A_{10} + A_8\gamma$。

S_v 为曲线的垂直方向漂移：$S_v = A_{11}F_z\gamma + A_{12}F_z + A_{13}$。

在纯侧偏角作用下轮胎的侧向力如图 2.8 所示

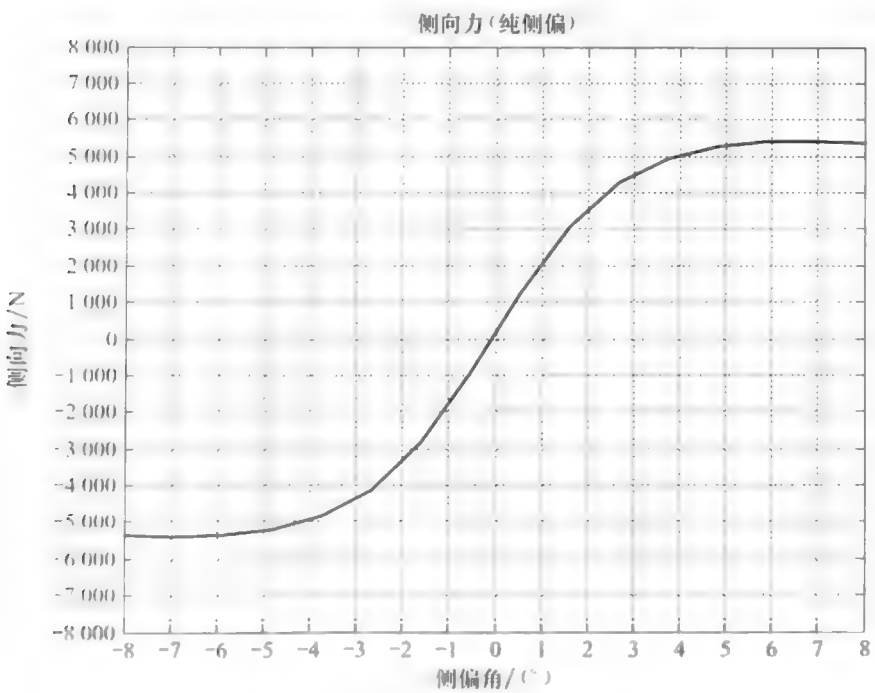


图 2.8 Pacejka'89 轮胎侧向力示例

（3）轮胎回正力矩计算

$$M_z = D\sin[C\arctan(Bx_1 - E(Bx_1 - \arctan(Bx_1)))] + S_v \tag{2.33}$$

此时的 x_1 为回正力矩计算组合自变量：$x_1=(\alpha+S_h)$，α 为侧偏角

C 为曲线形状因子，回正力矩计算时取 C_0 值：$C=C_0$

D 为巅因子，表示曲线的最大值：$D=C_1F_z^2+C_2F_z$

BCD 为回正力矩零点处的扭转刚度：$BCD=(C_3F_z^2+C_4F_z)\times(1-C_6|\gamma|)\times e^{-C_5F_z}$。

B 为刚度因子：$B=BCD/(C\times D)$。

S_h 为曲线的水平方向漂移：$S_h=C_{11}\gamma+C_{12}F_z+C_{13}$。

S_v 为曲线的垂直方向漂移：$S_v=\gamma(C_{14}F_z^2+C_{15}F_z)+C_{16}F_z+C_{17}$。

E 为曲线曲率因子，$E=(C_7F_z^2+C_8F_z+C_9)\times(1-C_{10}|\gamma|)$

在纯侧偏角作用下轮胎的回正力矩如图 2.9 所示。

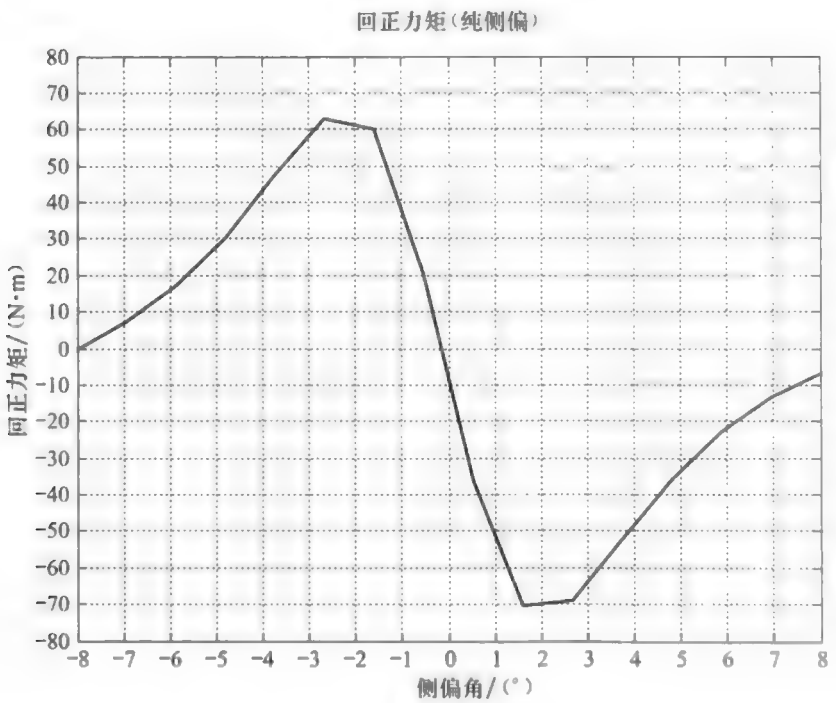


图 2.9　Pacejka'89 轮胎回正力矩示例

（4）Matlab m 文件实现

在轮胎纵向力和侧向力计算式中，系数 $a0-a13$，$b0-b10$，$c0-c17$ 根据轮胎实验数据拟合得到。本文所采用的系数如表 2-1 所示。

表 2-1 魔术公式参数的取值

参数名称	参数取值	参数名称	参数取值	参数名称	参数取值
a0	1.65	b0	2.372 72	c0	2.340 00
a1	-34	b1	-9.46	c1	1.495 0
a2	1 250	b2	1 490	c2	6.416 654
a3	3 036	b3	130	c3	-3.574 03
a4	12.8	b4	276	c4	-0.087 737
a5	0.005 01	b5	0.088 6	c5	0.098 410
a6	-0.021 03	b6	0.004 02	c6	0.002 769 9
a7	0.773 94	b7	0.061 5	c7	-0.000 115 1
a8	0.002 289 0	b8	1.2	c8	0.100 0
a9	0.013 442	b9	0.029 9	c9	-1.333 29
a10	0.003 70	b10	-0.176	c10	0.025 501
a11	19.165 6			c11	-0.023 57
a12	1.213 56			c12	0.030 27
a13	6.262 06			c13	-0.064 7
				c14	0.021 132 9
				c15	0.894 69
				c16	-0.099 443
				c17	-3.336 941

Matlab 仿真程序如下（注意在下面及本书后面的代码中，为保证与下载的源代码一致，说明与符号均采用正体）：

```
% Pacejka'89 轮胎模型认为轮胎在垂直、侧向方向上是线性的,阻尼为常量
% 这在侧向加速度常见范围≤0.4g、侧偏角≤5°的情景下对常规轮胎具有很高的拟合精度
% 这个算式里没有考虑地面摩擦系数的影响
%%
Fz=5;                                    % 垂直载荷 单位:kN
% Longitudinal Force(pure longitudinal slip)
% input:slip ratio 滑移率 s
s=linspace(-20,20,40);  % -20 20 插值 40 个,生成滑移率 s 横坐标
% ************ longitudinal coefficients 纵向系数 ******* %
```

```
    b0 = 2.37272;
    b1 = -9.46;
    b2 = 1490;
    b3 = 130;
    b4 = 276;
    b5 = 0.0886;
    b6 = 0.00402;
    b7 = -0.0615;
    b8 = 1.2;
    b9 = 0.0299;
    b10 = -0.176;
    % ********** parameters 参数 ********** %
    Cx = b0;                                    % 曲线形状因子
    Dx = b1 * Fz^2 + b2 * Fz;                   % 曲线巅因子
    BCDx = (b3 * Fz^2 + b4 * Fz) * exp(- b5 * Fz);
                                                % 纵向力零点处的纵向刚度
    Bx = BCDx/(Cx * Dx);                        % 刚度因子
    Shx = b9 * Fz + b10;                        % 曲线的水平方向漂移
    kx = s + Shx;                               % 输入变量 X1
    Svx = 0;                                    % 曲线的垂直方向漂移
    Ex = b6 * Fz^2 + b7 * Fz + b8;              % 曲线的曲率因子
    Fx = Dx * sin(Cx * atan(Bx * kx - Ex * (Bx * kx - atan(Bx * kx))))+
Svx;                                            % 纵向力的计算
    figure(1);
    plot(s,Fx);                                 % 绘制纵向力曲线
    grid  on;                                   % 绘制栅格
    set(gca,'xlim',[-20 20]);                   % 设置 x 轴范围
    set(gca,'xtick',[-20:5:20]);                % 设置 x 轴间隔
    set(gca,'ylim',[-8000 8000]);               % 设置 x 轴范围
    set(gca,'ytick',[-8000:1000:8000]);% 设置 x 轴间隔
    xlabel('纵向滑移率');
    ylabel('纵向力/(N)');
    title('纵向力(纯纵滑)');
    % Lateral Force(pure side slip)
```

```
% input:横向侧偏是侧偏角 alpha
alpha = linspace(-8,8,16);  % -8 ~ 8 插值 16 个,生成侧偏角横
坐标
r = 0;                                  % 外倾角,设为零
% ************ lateral coefficients ******************** %
a0 = 1.65;
a1 = -34;
a2 = 1250;
a3 = 3036;
a4 = 12.8;
a5 = 0.00501;
a6 = -0.02103;
a7 = 0.77394;
a8 = 0.0022890;
a9 = 0.013442;
a10 = 0.003709;
a11 = 19.1656;
a12 = 1.21356;
a13 = 6.26206;
% ********** parameters ********************** %
Cy = a0;                                % 曲线形状因子
Dy = a1 * Fz^2 + a2 * Fz;               % 曲线巅因子
BCDy = a3 * sin(2 * atan(Fz/a4)) * (1 - a5 * abs(r));
                                        % 侧向力零点处的侧向刚度
By = BCDy/(Cy * Dy);                    % 刚度因子
Shy = a9 * Fz + a10 + a8 * r;           % 曲线的水平方向漂移
ky = alpha + Shy;                       % 输入变量 X
Svy = a11 * Fz * r + a12 * Fz + a13;    % 曲线的垂直方向漂移
Ey = a6 * Fz^2 + a7;                    % 曲线曲率因子
% ******************** lateral force formulation
Fy0 = Dy * sin(Cy * atan(By * ky - Ey * (By * ky - atan(By * ky))))+
Svy;                                    % 纵向力的计算
figure(2);
plot(alpha,Fy0);
```

```
grid
set(gca,'xlim',[-8 8]);                    % 设置x轴范围
set(gca,'xtick',[-8:1:8]);                 % 设置x轴间隔
set(gca,'ylim',[-8000 8000]);              % 设置x轴范围
set(gca,'ytick',[-8000:1000:8000]);        % 设置x轴间隔
xlabel('侧偏角');
ylabel('侧向力/(N)');
title('侧向力(纯侧偏)');
%% Aligning Torque(pure side slip)
% input:侧偏角
%  ********************* ALIGNING_COEFFICIENTS ********** %
c0 =2.34000;
c1 =1.4950;
c2 =6.416654;
c3 =-3.57403;
c4 =-0.087737;
c5 =0.098410;
c6 =0.0027699;
c7 =-0.0001151;
c8 =0.1000;
c9 =-1.33329;
c10 =0.025501;
c11 =-0.02357;
c12 =0.03027;
c13 =-0.0647;
c14 =0.0211329;
c15 =0.89469;
c16 =-0.099443;
c17 =-3.336941;
%  ********** parameters ********************** %
Cz = c0;                                   % 曲线形状因子
Dz = c1 * Fz^2 + c2 * Fz;                  % 曲线峰因子
BCDz = (c3 * Fz^2 + c4 * Fz) * (1 - c5 * abs(r)) * exp(- c5 * Fz);
                                           % 回正力矩零点处的扭转
```

```
                                                    刚度
    Bz = BCDz/(Cz * Dz);                            % 刚度因子
    Shz = c11 * r + c12 * Fz + c13;                 % 曲线的水平方向漂移
    kz = alpha + Shz;                               % 输入变量 X
    Svz = r * (c14 * Fz^2 + c15 * Fz)+ c16 * Fz + c17;
                                                    % 曲线的垂直方向漂移
    Ez =(c7 * Fz^2 + c8 * Fz + c9) * (1 - c10 * abs(r));
                                                    % 曲线曲率因子
    % ******************** aligning torque formulation
    Mz0 = Dz * sin(Cz * atan(Bz * kz - Ez * (Bz * kz - atan(Bz * kz))))+
Svz;                                                % 纵向力的计算
    figure(3);
    plot(alpha,Mz0);
    grid
    set(gca,'xlim',[-8 8]);                         % 设置 x 轴范围
    set(gca,'xtick',[-8:1:8]);                      % 设置 x 轴间隔
    set(gca,'ylim',[-80 80]);                       % 设置 x 轴范围
    set(gca,'ytick',[-80:10:80]);                   % 设置 x 轴间隔
    xlabel('侧偏角');
    ylabel('回正力矩/(N)');
    title('回正力矩(纯侧偏)');
```

2.2.3 小角度假设下的车辆动力学模型

通过将式（2.29）和轮胎模型结合，可以发现建立的非线性状态空间表达式相对于模型预测控制器的设计来说还是过于复杂，因此需要进一步简化。图 2.10 为根据实验数据所绘制的不同载荷下轮胎纵向力及侧向力曲线。从图 2.10 中可以看出，在侧偏角及纵向滑移率较小时，轮胎力可以用线性函数近似描述。这在侧向加速度 $a_y \leqslant 0.4\ g$ 的情况下对常规轮胎具有较高的拟合精度。在这个范围内，用以下算式得到轮胎的纵向力和侧向力：

$$F_x = C_s s \quad F_y = C_\alpha \alpha \tag{2.34}$$

式中，C_s 为轮胎纵向刚度，C_α 为轮胎侧偏刚度。

在之前所建立的非线性模型中，存在较多的三角函数，对于模型简化有较

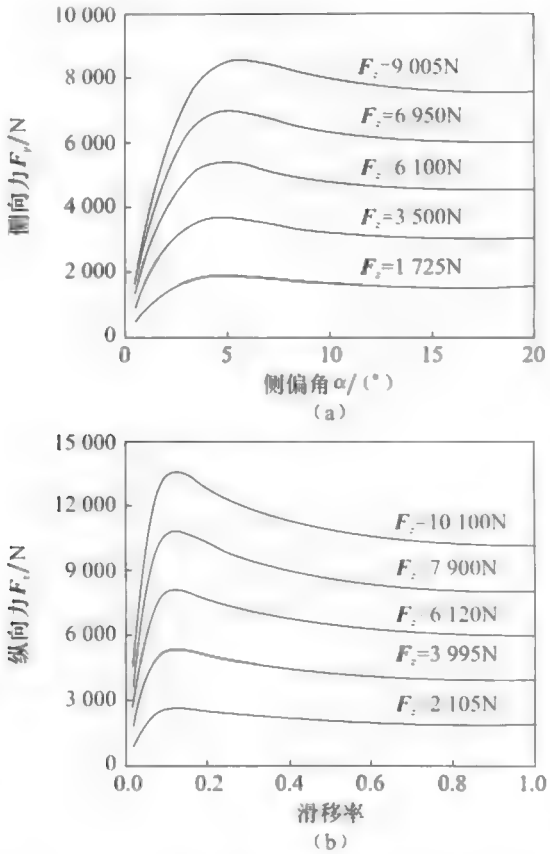


图 2.10　轮胎纵向力及侧向力曲线

（a）侧向力随侧偏角变化曲线；（b）纵向力随滑移率变化曲线

大的困难。因此，在轮胎力的计算过程中，提出小角速度假设，即满足如下近似条件：

$$\cos\theta \approx 1, \quad \sin\theta \approx \theta, \quad \tan\theta \approx \theta$$

式中，θ 表示各个角，包括前轮偏角、前、后轮胎侧偏角等

通过以上简化并结合式（2.19）~式（2.23），可以得到轮胎侧偏角的计算关系式：

$$\alpha_f = \frac{\dot{y} + a\dot{\varphi}}{\dot{x}} - \delta_f \quad \alpha_r = \frac{\dot{y} - b\dot{\varphi}}{\dot{x}} \tag{2.35}$$

根据式（2.35）和线性轮胎模型，就可以很容易计算得到轮胎力

前、后轮胎的侧向力：

$$F_{cf} = C_{cf}\left(\delta_f - \frac{\dot{y} + a\dot{\varphi}}{\dot{x}}\right) \quad F_{cr} = C_{cr}\frac{b\dot{\varphi} - \dot{y}}{\dot{x}} \tag{2.36}$$

前、后轮胎的纵向力：

$$F_{\mathrm{lf}} = C_{\mathrm{lf}} s_{\mathrm{f}} \quad F_{\mathrm{lr}} = C_{\mathrm{lr}} s_{\mathrm{r}} \tag{2.37}$$

将以上简化后的结果代入式（2.29）后，得到了基于前轮偏角较小和线性轮胎模型假设后的车辆动力学非线性模型：

$$\begin{aligned}
m\ddot{y} &= -m\dot{x}\dot{\varphi} + 2\left[C_{\mathrm{cf}}\left(\delta_{\mathrm{f}} - \frac{\dot{y} + a\dot{\varphi}}{\dot{x}}\right) + C_{\mathrm{cr}}\frac{b\dot{\varphi} - \dot{y}}{\dot{x}}\right] \\
m\ddot{x} &= m\dot{y}\dot{\varphi} + 2\left[C_{\mathrm{lf}} s_{\mathrm{f}} + C_{\mathrm{cf}}\left(\delta_{\mathrm{f}} - \frac{\dot{y} + a\dot{\varphi}}{\dot{x}}\right)\delta_{\mathrm{f}} + C_{\mathrm{lr}} s_{\mathrm{r}}\right] \\
I_z\ddot{\varphi} &= 2\left[aC_{\mathrm{cf}}\left(\delta_{\mathrm{f}} - \frac{\dot{y} + a\dot{\varphi}}{\dot{x}}\right) - bC_{\mathrm{cr}}\frac{b\dot{\varphi} - \dot{y}}{\dot{x}}\right] \\
\dot{Y} &= \dot{x}\sin\varphi + \dot{y}\cos\varphi \\
\dot{X} &= \dot{x}\cos\varphi - \dot{y}\sin\varphi
\end{aligned} \tag{2.38}$$

在该系统中，状态量选取为 $\boldsymbol{\xi}_{\mathrm{dyn}} = [\dot{y},\ \dot{x},\ \varphi,\ \dot{\varphi},\ Y,\ X]^{\mathrm{T}}$，控制量选取为 $\boldsymbol{u}_{\mathrm{dyn}} = \delta_{\mathrm{f}}$

第3章

模型预测控制算法基础与仿真分析

在初学模型预测控制时往往会感觉比较抽象。模型预测控制作为一种控制方法，它的简单在于思想非常贴近我们的日常生活，通过一个小小的例子就能明白它的核心内容，但要深入研究模型预测控制方法，则存在大量的数学基础问题和控制理论问题，在数学计算复杂性、控制实时性和稳定性问题等方面，还有大量的问题需要我们去解决和探索。

本章首先介绍模型预测控制基础理论，然后扩展到两类常用的模型预测控制方法。为了避免在开始学习时就陷入深奥的理论中，在介绍理论时，首先，从日常生活中的一个案例引出模型预测控制的基本思想；其次，通过一个工程中的实例进一步加深对这种方法的认识；再次，分别对线性模型预测控制和非线性模型预测控制进行介绍，从代码分析的角度出发，让读者能够尽快编写自己的控制算法。

3.1 基本理论

3.1.1 生活中的启示

情景如下：团队接到一项任务，要求用8 h完成。根据我们已有的经验，我们知道这项任务可以分解成8个子任务，并且1 h大致能够完成一项（注意：并不是一定可以完成）。那么，我们应该如何规划这8 h呢？也许很多人

会第一时间想到图 3.1 中所示的方案：既然一个小时可以完成一项任务，那肯定就是每个小时完成一项子任务，8 h 结束后自然就可以完成所有的任务了。这个方案虽然具备一定的可行性，但具体执行时还是会遇到很多问题，比如团队成员临时离开、某一个子任务没有按时完成，等等。因此，这个方案在实际执行过程中过于理想化。

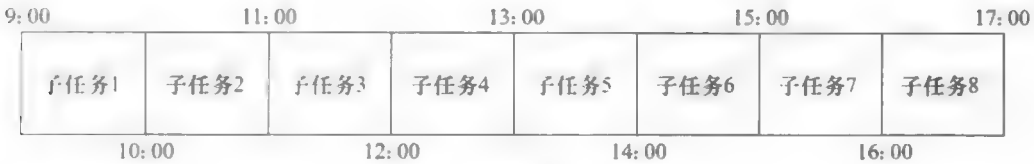


图 3.1　工作规划方案 1

为了解决这些突发性问题，提高整个工作过程的可行性，经过思考，我们可以提出第 2 种方案，也就是图 3.2 所示的方案。在这个方案中，我们不再规划所有任务，而是规划出未来一段时间（比如 4 h）内，依然是每个小时完成其中一项任务。当然，在这一段时间内，依然存在发生突发情况的可能。1 h 过去后，还需要对已经完成的任务进行总结。在新的 1 h 开始时，首先需要根据过去完成的结果重新对未来 4 h 进行任务规划。如果上一个小时没有完成规定的任务，就会出现如图 3.3 所示的规划方案。在下一个时刻，就需要既完成上一时刻遗留下来的任务，也需要完成本时刻的任务。根据这样的规律，我们依次在新的时刻规划未来 4 h 的工作计划，并及时更正这样的计划，最终完成任务。

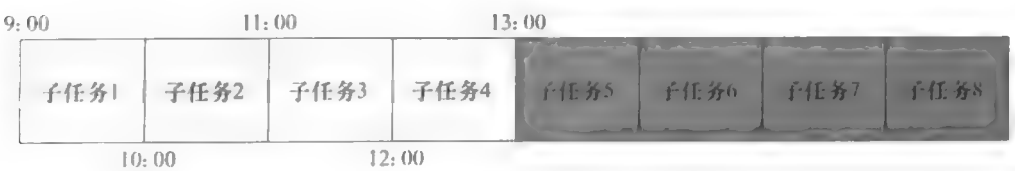


图 3.2　工作规划方案 2

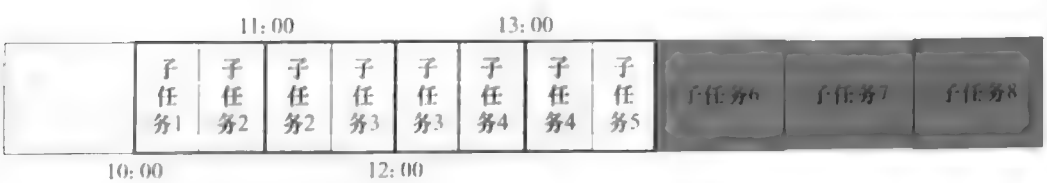


图 3.3　下一时刻新的规划方案

显然，方案2在实际工作中更加具备可操作性，它能根据任务完成情况及时调整工作计划，而较短的规划周期也使得出现突发情况的概率更小。在完成任务的过程中，我们可以总结出以下3个关键步骤：

① 根据已有的知识和经验可以预计任务的完成情况，如在上面的例子中，团队以往完成任务的能力就是已有的经验模型。

② 只规划未来固定一段时间的任务计划，而不是整个工作周期。

③ 每一固定时间段结束后检查任务完成情况，然后根据现有状态重新规划下一段时间的任务计划。

实际上，模型预测控制的基本思想就蕴含在这3个步骤中。它利用一个已有的模型、系统当前的状态和未来的控制量去预测系统未来的输出（步骤1），这个输出的长度是控制周期的整数倍（步骤2）。由于未来的控制量是未知的，还需要根据一定的优化条件进行优化求解，以得到未来的控制量序列。在每一个控制周期结束后，系统根据当前实际状态重新预测系统未来输出（步骤3）。关于这3个关键步骤的进一步介绍将在下一节展开。

3.1.2 控制理论中的描述

根据上一节，模型预测控制在实现过程中有3个关键步骤，一般被称为3项基本原理，分别是预测模型、滚动优化和反馈校正[66]。以下分别对这3项基本原理进行介绍。

① 预测模型：预测模型是模型预测控制的基础，其主要功能是根据对象的历史信息和未来输入，预测系统未来的输出。对预测模型的形式没有做严格的限定，状态方程、传递函数这类传统的模型都可以作为预测模型。对于线性稳定系统，阶跃响应、脉冲响应这类非参数模型，也可以直接作为预测模型使用。

② 滚动优化：模型预测控制通过某一性能指标的最优来确定控制作用，但优化不是一次离线进行，而是反复在线进行的。这就是滚动优化的含义，也是模型预测控制区别于传统最优控制的根本点。

③ 反馈校正：为了防止模型失配或者环境干扰引起控制对理想状态的偏离，在新的采样时刻，首先检测对象的实际输出，并利用这一实时信息对基于模型的预测结果进行修正，然后再进行新的优化。

基于这3个要素，模型预测控制的基本原理可以用图3.4来表示。控制过程中，始终存在一条期望参考轨迹，如图3.4中曲线1所示。以时刻k作为当前时刻（坐标系纵轴所在位置），控制器结合当前的测量值和预测模型，预测

系统未来一段时域内 $[k, k+N_p]$（也被称为预测时域）系统的输出，如图 3.4 中曲线 2 所示。通过求解满足目标函数以及各种约束的优化问题，得到在控制时域 $[k, k+N_c]$ 内一系列的控制序列，如图中的矩形波 4 所示（从坐标系纵轴开始），并将该控制序列的第一个元素作为受控对象的实际控制量，当来到下一个时刻 $k+1$ 时，重复上述过程，如此滚动地完成一个个带约束的优化问题，以实现对被控对象的持续控制。

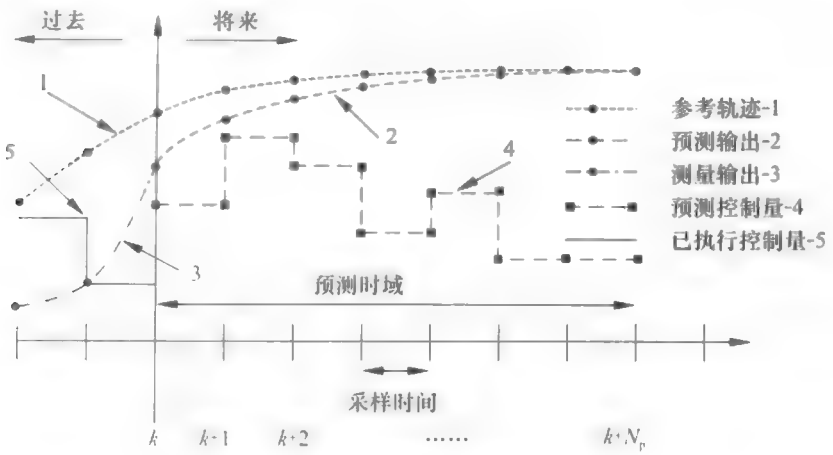


图 3.4　模型预测控制原理示意

模型预测控制原理框图如图 3.5 所示，包含了 MPC 控制器、被控平台和状态估计器这 3 个模块。其中，MPC 控制器结合预测模型、目标函数和约束条件进行最优化求解，得到当前时刻的最优控制序列 $\boldsymbol{u}^*(t)$，输入到被控平台，被控平台按照当前的控制量进行控制，然后将当前的状态量观测值 $\boldsymbol{x}(t)$ 输入给状态估计器。状态估计器对于那些无法通过传感器观测得到或者观测成本过高的状态量进行估计。比较常用的方法有 Kalman 滤波、粒子滤波等。将估计的状态量 $\hat{\boldsymbol{x}}(t)$ 输入到 MPC 控制器，再次进行最优化求解，以得到未来一段时间的控制序列。如此循环，就构成了完整的模型预测控制过程。

根据所采用模型的不同，模型预测控制主要包括动态矩阵控制（Dynamic Matrix Control，DMC）、模型算法控制（Model Algorithm Control，MAC）、广义预测控制（Generalized Predictive Control，GPC）等。同时，在现代控制理论中广泛使用的状态空间模型，同样可以应用于模型预测控制中。本书立足于模型预测控制在无人驾驶车辆方向的应用，因此重点放在基于状态空间模型的模型预测控制上。

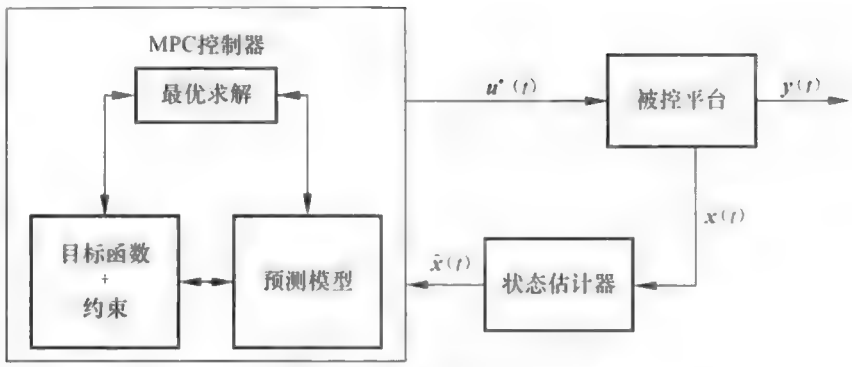


图 3.5　MPC 控制原理框图

3.2　一个简单的实例

通过第 3.1 节中的引例和理论介绍，对模型预测控制的 3 项基本原理有了一定的认识。下面通过在 Matlab 中的一个简单实例，分析模型预测控制在实际应用时的基本思路。

Matlab 中的模型预测控制工具箱（Model Predictive Control Toolbox）是集成在 Matlab 环境下的工具包之一，提供了一系列用于模型预测控制的分析、设计和仿真的函数。这个工具箱使用简单，能够帮助初学者快速了解 MPC 的基本概念和应用方法；同时，工具箱还提供了一系列实际控制例程作为教学使用。我们在开始学习模型预测控制时，就可以从这些案例出发

首先需要打开 Matlab 的例程教学窗口。在 Windows 中运行 Matlab 主程序，并在控制命令窗口输入“demo”命令，出现如图 3.6 所示界面　在左侧目录中可以看到众多工具箱，如神经网络工具箱、模糊工具箱等　找到 Model Predictive Control Toolbox，并将该项目展开，可以看到一系列子项目，其中 Demos 就是需要用到的例程。点击 Demos 下的 Tutorial 并选择第一个项目“MPC Control of Double Integrator”，可在右侧窗口看到这个例程的介绍、源代码以及结果展示。

为了直观地了解这个例程的功能，可以运行它查看结果　一般有两种方法：一种就是点击右上角的“Run in the Command Window”，如图 3.7 中黑色矩形框所示。进入 Matlab 控制界面后按照提示就可以看到每一步的运行过程。

另外一种方法是通过运行相应的 m 文件，点击主窗口左上角的“Open mpcdoubleint. m in the Editor”，如图 3.8 中黑色矩形框所示　然后程序会自

图 3.6　Matlab 例程教学窗口

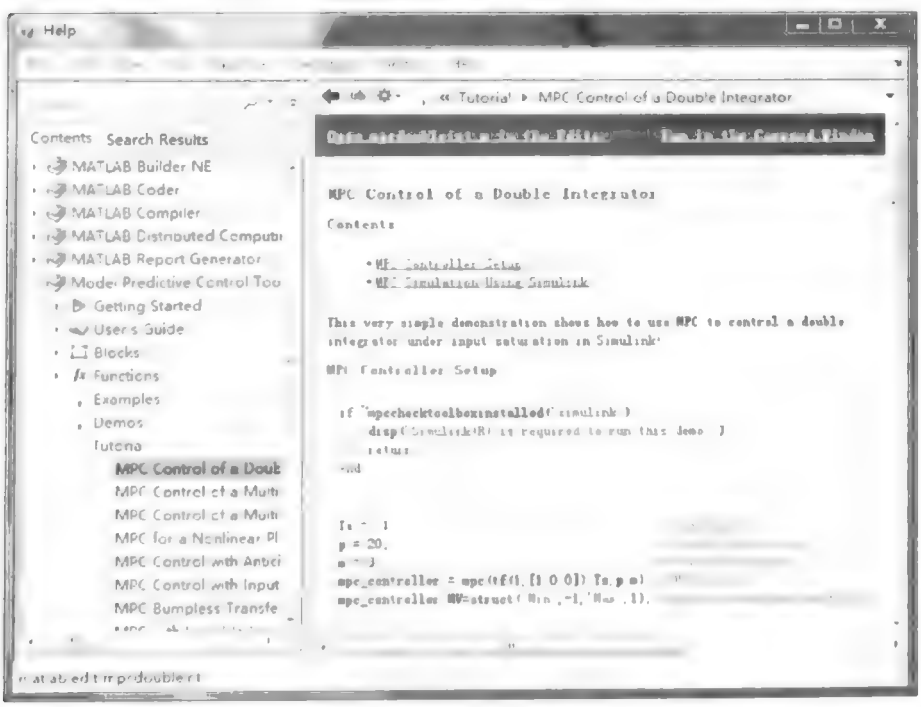


图 3.7　运行例程方法 1

图 3.8　运行例程方法 2

动打开相应的 m 文件，点击运行按钮或者键盘 F5，都可以直接运行这个程序。经过一段时间的计算后，程序会显示运行结果并且自动关闭窗口。如果要仔细观察控制结果，可以将 m 文件的最后两行程序注释掉，即：

```
% bdclose('mpc_doubleint');
% displayEndOfDemoMessage(mfilename)
```

运行后结果如图 3.9 所示。它包含 3 个窗口，左侧窗口为 Simulink 环境下搭建的控制系统，右侧两个窗口分别为被控系统的输入和输出。从被控系统输出我们可以看出，该系统的控制目标是跟踪一条固定轨迹，系统状态从 0 出发，在控制器的作用下，逐渐跟踪上了期望轨迹。从被控系统输入可以看出，输入值被成功限定在［-1，1］区间。下面介绍仿真实例中 MPC 控制系统如何实现模型预测控制功能。

首先，对整个控制系统进行分析。除去输入、输出显示，整个控制器主要由两部分内容组成：模型预测控制器和被控系统（双积分系统），如图 3.10

所示。具体的输入、输出关系如下：模型预测控制器的输入为参考轨迹和系统状态，而被控对象的输入则为控制器输出的控制量，被控对象的输出即状态量，再反馈输入到 MPC 控制器中。明确了输入、输出关系后能够为后续自行设计控制系统奠定基础。

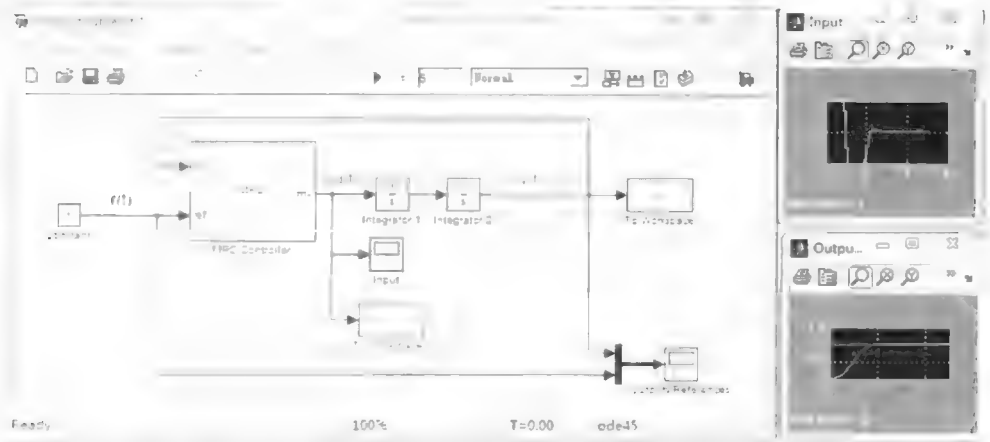


图 3.9　运行结果

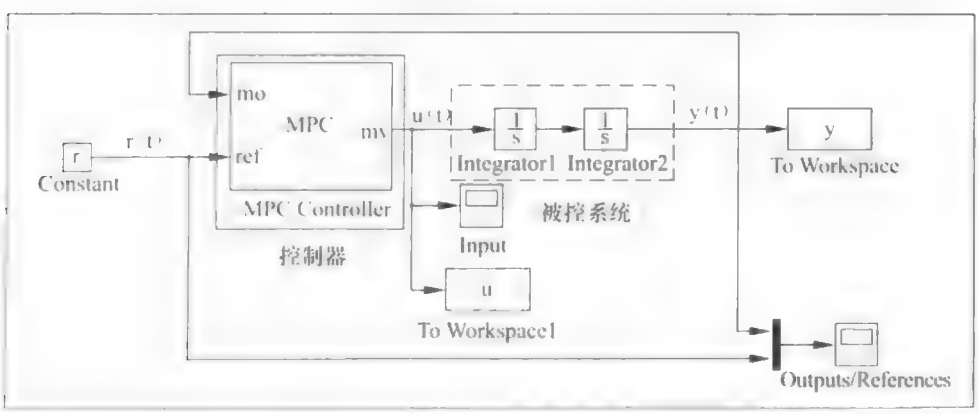


图 3.10　控制系统分析

在对整个控制系统有了基本概念后，再来了解算法程序代码，探究一个标准的模型预测控制器究竟如何工作。在之前打开的“mpcdoubleint. m”文件中，首先是对程序的说明以及版权申明，然后按照不同的步骤分块进行程序编写。今后我们在编写自己的控制算法时，也可以参考这样的写法。在程序主体内容中，我们将重点放在模型预测控制器的设置上面，也就是如下一段代码：

```
Ts = .1;                          % Sampling time
p = 20;                           % Prediction horizon
m = 3;                            % Control horizon
mpc_controller = mpc(tf(1,[1 0 0]),Ts,p,m);
                                  % MPC object
mpc_controller.MV = struct('Min',-1,'Max',1);
                                  % Input saturation constraints
```

这段代码完成了构建模型预测控制器必要的设置。首先是对系统采样时间的设置；其次是确定了预测时域和控制时域（这两者的区别将在之后的章节中介绍）；再次是通过调用系统函数 tf( ) 以传递函数的形式实现系统建模；最后通过调用模型预测工具箱函数 mpc( ) 完成控制器的设定。为了实现对控制输入的限制，代码还加入了控制量的上下界约束条件。

一般来说，模型预测控制器需要给定被控系统的采样时间、预测时域、控制时域和预测模型。如果需要加入系统的约束，还需要进一步给定约束条件。不仅仅是在 Matlab 自带的工具箱中，其他版本的工具箱（如 MPC tools）也是在此基础上扩展的。读者在编写控制算法的时候，也建议将这些参数作为基本的传递参数。这样不仅能保证设置准确可靠，还能提高代码的可读性。

由于是在 Simulink 环境下的仿真，故还需要对仿真环境进行设置，包括系统初始状态、仿真时间和参考轨迹，代码如下：

```
x01 = 0;              % Initial state:First integrator
x02 = 0;              % Initial state:Second integrator
Tstop = 5;            % Simulation time
r = 1;                % Set point
```

最后，打开搭建的 Simulink 模型，运行仿真程序，就完成了整个控制系统。

分析了整个程序之后，会发现这个例子其实非常简单。通过简单的参数设置和环境设置，就可以完成一个双积分系统对固定轨迹的跟踪，并且对输入、输出进行了约束设定。为了加深对程序的理解，我们可以自己尝试修改相关参数，如预测时域、采样时间等，观看不同参数的控制效果。在对模型预测控制有了一定的认识后，可以分别修改控制器与被控对象，搭建自己的仿真平台。

通过模型预测控制工具箱中的例程，对模型预测控制如何使用已经有了直

观的认识；但在刚才的分析中，只了解了如何去设置模型预测控制器的参数，但并没有涉及模型预测控制器的内部结构。打开工具箱函数 mpc( ) 的 m 文件，发现程序比较复杂，一时间难以看懂。这时候就需要我们掌握将模型预测控制问题转换为计算机算法程序应用的方法，同时还要补充一些控制系统的知识。只有建立在良好的理论基础之上，才能开始编写核心的控制算法。

3.3　线性时变模型预测控制算法

3.3.1　问题描述

线性时变模型预测控制算法是以线性时变模型作为预测模型的。这也是目前在模型预测控制领域中应用最为广泛的一种形式。相比于非线性模型预测控制，它的最大优点是计算较为简单，实时性好。对于无人驾驶车辆的运动控制来说，控制算法的实时性显然是至关重要的，因此线性时变模型预测控制是需要重点介绍的一种 MPC 控制方法。

在之前的介绍中，已经反复强调了模型预测控制器的 3 项基本原理。以下，以线性状态空间模型为基础，依次推导模型预测控制的预测方程、优化求解以及反馈机制。

（1）预测方程

首先，考虑以下的离散线性化模型：

$$\boldsymbol{x}(k+1) = \boldsymbol{A}_{k,t}\boldsymbol{x}(k) + \boldsymbol{B}_{k,t}\boldsymbol{u}(k) \tag{3.1}$$

设定

$$\boldsymbol{\xi}(k \mid t) = \begin{bmatrix} \boldsymbol{x}(k \mid t) \\ \boldsymbol{u}(k-1 \mid t) \end{bmatrix} \tag{3.2}$$

可以得到一个新的状态空间表达式：

$$\begin{aligned} \boldsymbol{\xi}(k+1 \mid t) &= \tilde{\boldsymbol{A}}_{k,t}\boldsymbol{\xi}(k \mid t) + \tilde{\boldsymbol{B}}_{k,t}\Delta\boldsymbol{u}(k \mid t) \\ \boldsymbol{\eta}(k \mid t) &= \tilde{\boldsymbol{C}}_{k,t}\boldsymbol{\xi}(k \mid t) \end{aligned} \tag{3.3}$$

式中各矩阵的定义如下：

$$\tilde{\boldsymbol{A}}_{k,t} = \begin{bmatrix} \boldsymbol{A}_{k,t} & \boldsymbol{B}_{k,t} \\ \boldsymbol{0}_{m\times n} & \boldsymbol{I}_m \end{bmatrix}, \quad \tilde{\boldsymbol{B}}_{k,t} = \begin{bmatrix} \boldsymbol{B}_{k,t} \\ \boldsymbol{I}_m \end{bmatrix} \tag{3.4}$$

$$\tilde{\boldsymbol{C}}_{k,t} = [\boldsymbol{C}_{k,t} \quad \boldsymbol{0}]$$

为了进一步简化计算，做出如下假设：

$$\begin{aligned}\tilde{\boldsymbol{A}}_{k,t} &= \tilde{\boldsymbol{A}}_t, \quad k = 1,\cdots,t+N-1\\ \tilde{\boldsymbol{B}}_{k,t} &= \tilde{\boldsymbol{B}}_t, \quad k = 1,\cdots,t+N-1\end{aligned} \tag{3.5}$$

如果系统的预测时域为 N_p，控制时域为 N_c，那么，预测时域内的状态量和系统输出量可用以下算式计算：

$$\boldsymbol{\xi}(t+N_p \mid t) = \tilde{\boldsymbol{A}}_t^{N_p}\boldsymbol{\xi}(t \mid t) + \tilde{\boldsymbol{A}}_t^{N_p-1}\tilde{\boldsymbol{B}}_t\Delta\boldsymbol{u}(t \mid t) + \cdots + \tilde{\boldsymbol{A}}_t^{N_p-N_c-1}\tilde{\boldsymbol{B}}_t\Delta\boldsymbol{u}(t+N_c \mid t) \tag{3.6}$$

$$\begin{aligned}\boldsymbol{\eta}(t+N_p \mid t) = {}&\tilde{\boldsymbol{C}}_{t,t}\tilde{\boldsymbol{A}}_{t,t}^{N_p}\boldsymbol{\xi}(t \mid t) + \tilde{\boldsymbol{C}}_t\tilde{\boldsymbol{A}}_t^{N_p-1}\tilde{\boldsymbol{B}}_t\Delta\boldsymbol{u}(t \mid t) + \cdots +\\ &\tilde{\boldsymbol{C}}_t\tilde{\boldsymbol{A}}_t^{N_p-N_c-1}\tilde{\boldsymbol{B}}_t\Delta\boldsymbol{u}(t+N_c \mid t)\end{aligned} \tag{3.7}$$

为了使整个关系更加明确，将系统未来时刻的输出以矩阵的形式表达：

$$\boldsymbol{Y}(t) = \boldsymbol{\Psi}_t\boldsymbol{\xi}(t \mid t) + \boldsymbol{\Theta}_t\Delta\boldsymbol{U}(t) \tag{3.8}$$

式中：

$$\boldsymbol{Y}(t) = \begin{bmatrix}\boldsymbol{\eta}(t+1 \mid t)\\ \boldsymbol{\eta}(t+2 \mid t)\\ \cdots\\ \boldsymbol{\eta}(t+N_c \mid t)\\ \cdots\\ \boldsymbol{\eta}(t+N_p \mid t)\end{bmatrix} \quad \boldsymbol{\Psi}_t = \begin{bmatrix}\tilde{\boldsymbol{C}}_t\tilde{\boldsymbol{A}}_t\\ \tilde{\boldsymbol{C}}_t\tilde{\boldsymbol{A}}_t^2\\ \cdots\\ \tilde{\boldsymbol{C}}_t\tilde{\boldsymbol{A}}_t^{N_c}\\ \cdots\\ \tilde{\boldsymbol{C}}_t\tilde{\boldsymbol{A}}_t^{N_p}\end{bmatrix} \quad \Delta\boldsymbol{U}(t) = \begin{bmatrix}\Delta\boldsymbol{u}(t \mid t)\\ \Delta\boldsymbol{u}(t+1 \mid t)\\ \cdots\\ \Delta\boldsymbol{u}(t+N_c \mid t)\end{bmatrix}$$

$$\boldsymbol{\Theta}_t = \begin{bmatrix}\tilde{\boldsymbol{C}}_t\tilde{\boldsymbol{B}}_t & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}\\ \tilde{\boldsymbol{C}}_t\tilde{\boldsymbol{A}}_t\tilde{\boldsymbol{B}}_t & \tilde{\boldsymbol{C}}_t\tilde{\boldsymbol{B}}_t & \boldsymbol{0} & \boldsymbol{0}\\ \cdots & \cdots & \ddots & \cdots\\ \tilde{\boldsymbol{C}}_t\tilde{\boldsymbol{A}}_t^{N_c-1}\tilde{\boldsymbol{B}}_t & \tilde{\boldsymbol{C}}_t\tilde{\boldsymbol{A}}_t^{N_c-2}\tilde{\boldsymbol{B}}_t & \cdots & \tilde{\boldsymbol{C}}_t\tilde{\boldsymbol{B}}_t\\ \tilde{\boldsymbol{C}}_t\tilde{\boldsymbol{A}}_t^{N_c}\tilde{\boldsymbol{B}}_t & \tilde{\boldsymbol{C}}_t\tilde{\boldsymbol{A}}_t^{N_c-1}\tilde{\boldsymbol{B}}_t & \cdots & \tilde{\boldsymbol{C}}_t\tilde{\boldsymbol{A}}_t\tilde{\boldsymbol{B}}_t\\ \vdots & \vdots & \ddots & \vdots\\ \tilde{\boldsymbol{C}}_t\tilde{\boldsymbol{A}}_t^{N_p-1}\tilde{\boldsymbol{B}}_t & \tilde{\boldsymbol{C}}_t\tilde{\boldsymbol{A}}_t^{N_p-2}\tilde{\boldsymbol{B}}_t & \cdots & \tilde{\boldsymbol{C}}_t\tilde{\boldsymbol{A}}_t^{N_p-N_c-1}\tilde{\boldsymbol{B}}_t\end{bmatrix}$$

通过观察式（3.8），可以清楚地看到，在预测时域内的状态量和输出量都可以通过系统当前的状态量 $\boldsymbol{\xi}(t \mid t)$ 和控制时域内的控制增量 $\Delta\boldsymbol{U}(t)$ 计算得到。这也就是模型预测控制算法中“预测”功能的实现。

（2）优化求解

实际上，系统的控制增量是未知的，只有通过设定合适的优化目标，并对其进行求解，才能得到控制时域内的控制序列。文献［50］给出了如下形式的目标函数：

$$J(k) = \sum_{j=1}^{N} \tilde{\boldsymbol{X}}^{\mathrm{T}}(k+j \mid k)\boldsymbol{Q}\tilde{\boldsymbol{X}}(k+j) + \tilde{\boldsymbol{u}}^{\mathrm{T}}(k+j-1)\boldsymbol{R}\tilde{\boldsymbol{u}}(k+j-1) \tag{3.9}$$

对于以上形式的优化目标，可以通过适当的处理将其转换为二次规划（Quadratic Programming，QP）问题。二次规划是一个典型的数学优化问题，它的优化目标是二次实函数，带有线性或者非线性约束。其常用的解法为有效集法（Active Set Method，ASM）和内点法（Interior Point Method，IPM）。有效集法适用于解决只有不等式约束的二次规划问题，而内点法适用于任何形式的二次规划问题。文献[50]已详细介绍了将模型预测控制问题转换为二次规划问题的方法，本章第3.5节给出了基于线性二次型最优控制的实例，并给出了算法代码。

式（3.9）中以控制量作为目标函数中的状态量，结构简单，易于实现；但也存在一些缺点，比较显著的就是没法对控制增量进行精确约束。当系统对于控制量跳变要求较为严格时，这样的目标函数就无能为力了。此时，可以把控制增量作为目标函数的状态量，优化目标函数可以设为如下形式：

$$J(\boldsymbol{\xi}(t),\boldsymbol{u}(t-1),\Delta\boldsymbol{U}(t)) = \sum_{i=1}^{N_p} \| \boldsymbol{\eta}(t+i \mid t) - \boldsymbol{\eta}_{\mathrm{ref}}(t+i \mid t) \|_{\boldsymbol{Q}}^{2} + \sum_{i=1}^{N_c-1} \| \Delta\boldsymbol{u}(t+i \mid t) \|_{\boldsymbol{R}}^{2} \tag{3.10}$$

其中，第一项反映了系统对参考轨线的跟随能力，第二项反映了对控制量平稳变化的要求。$\boldsymbol{Q}$ 和 $\boldsymbol{R}$ 为权重矩阵，整个表达式的功能是使系统能够尽快且平稳地跟踪上期望的轨迹。同时，在实际的控制系统中，往往需要满足系统状态量以及控制量的一些约束条件，一般如下：

控制量约束：

$$\boldsymbol{u}_{\min}(t+k) \leqslant \boldsymbol{u}(t+k) \leqslant \boldsymbol{u}_{\max}(t+k), \quad k = 0,1,\cdots,N_c-1 \tag{3.11}$$

控制增量约束：

$$\Delta\boldsymbol{u}_{\min}(t+k) \leqslant \Delta\boldsymbol{u}(t+k) \leqslant \Delta\boldsymbol{u}_{\max}(t+k), \quad k = 0,1,\cdots,N_c-1 \tag{3.12}$$

输出约束：

$$\boldsymbol{y}_{\min}(t+k) \leqslant \boldsymbol{y}(t+k) \leqslant \boldsymbol{y}_{\max}(t+k), \quad k = 0,1,\cdots,N_c-1 \tag{3.13}$$

式（3.10）~（3.13）就形成了一个完整的优化目标表达式。通过求解这个带约束条件的优化目标，就能得到未来一段时间的控制序列。然而，由于系统的模型是实时改变的，并不能保证每个时刻该优化目标都能得到可行解。因

此，有必要对优化目标进行相应的处理。比较普遍并且证明有效的方式是在优化目标中加入松弛因子，如式（3.14）所示：

$$J(\boldsymbol{\xi}(t),\boldsymbol{u}(t-1),\Delta\boldsymbol{U}(t)) = \sum_{i=1}^{N_p} \| \boldsymbol{\eta}(t+i\mid t) - \boldsymbol{\eta}_{ref}(t+i\mid t) \|_{\boldsymbol{Q}}^2 + \sum_{i=1}^{N_c-1} \| \Delta\boldsymbol{u}(t+i\mid t) \|_{\boldsymbol{R}}^2 + \rho\varepsilon^2 \tag{3.14}$$

式中，ρ 为权重系数，ε 为松弛因子。

将式（3.8）代入优化目标式（3.14），并且将预测时域内的输出量偏差表示为：

$$\boldsymbol{E}(t) = \boldsymbol{\Psi}_t\tilde{\boldsymbol{\xi}}(t\mid t) - \boldsymbol{Y}_{ref}(t),\quad \boldsymbol{Y}_{ref} = [\boldsymbol{\eta}_{ref}(t+1\mid t),\cdots,\boldsymbol{\eta}_{ref}(t+N_p\mid t)]^{\mathrm{T}} \tag{3.15}$$

经过相应的矩阵计算，可以将优化目标调整为：

$$J(\boldsymbol{\xi}(t),\boldsymbol{u}(t-1),\Delta\boldsymbol{U}(t)) = [\Delta\boldsymbol{U}(t)^{\mathrm{T}},\varepsilon]^{\mathrm{T}}\boldsymbol{H}_t[\Delta\boldsymbol{U}(t)^{\mathrm{T}},\varepsilon] + \boldsymbol{G}_t[\Delta\boldsymbol{U}(t)^{\mathrm{T}},\varepsilon] + P_t \tag{3.16}$$

式中：$\boldsymbol{H}_t = \begin{bmatrix} \boldsymbol{\Theta}_t^{\mathrm{T}}\boldsymbol{Q}_e\boldsymbol{\Theta}_t + \boldsymbol{R}_c & \boldsymbol{0} \\ \boldsymbol{0} & \rho \end{bmatrix}$，$\boldsymbol{G}_t = [2\boldsymbol{E}(t)^{\mathrm{T}}\boldsymbol{Q}_e\boldsymbol{\Theta}_t \quad 0]$，$P_t = \boldsymbol{E}(t)^{\mathrm{T}}\boldsymbol{Q}_e\boldsymbol{E}(t)$

在式（3.16）中，P_t 为常量，因此模型预测控制在每一步的带约束优化求解问题都等价于求解如下的二次规划问题：

$$\begin{aligned} &\min_{\Delta\boldsymbol{U}_t,\varepsilon}[\Delta\boldsymbol{U}(t)^{\mathrm{T}},\varepsilon]^{\mathrm{T}}\boldsymbol{H}_t[\Delta\boldsymbol{U}(t)^{\mathrm{T}},\varepsilon] + \boldsymbol{G}_t[\Delta\boldsymbol{U}(t)^{\mathrm{T}},\varepsilon] \\ &\text{s.t.}\quad \Delta\boldsymbol{U}_{\min} \leqslant \Delta\boldsymbol{U}(k) \leqslant \Delta\boldsymbol{U}_{\max},\quad k = t,\cdots,t+H_c-1 \\ &\boldsymbol{U}_{\min} \leqslant \boldsymbol{u}(t-1) + \sum_{i=t}^{k}\Delta\boldsymbol{U}(i) \leqslant \boldsymbol{U}_{\max},\quad k = t,\cdots,t+H_c-1 \\ &\boldsymbol{Y}_{\min} - \varepsilon \leqslant \boldsymbol{\Psi}_t\boldsymbol{\xi}(t\mid t) + \boldsymbol{\Theta}_t\Delta\boldsymbol{U}(t) \leqslant \boldsymbol{Y}_{\max} + \varepsilon \\ &\varepsilon > 0 \end{aligned} \tag{3.17}$$

（3）反馈机制

在每个控制周期内完成对式（3.17）的求解后，得到了控制时域内的一系列控制输入增量：

$$\Delta\boldsymbol{U}_t^* = [\Delta\boldsymbol{u}_t^*,\Delta\boldsymbol{u}_{t+1}^*,\cdots,\Delta\boldsymbol{u}_{t+N_c-1}^*]^{\mathrm{T}} \tag{3.18}$$

根据模型预测控制的基本原理，将该控制序列中第一个元素作为实际的控制输入增量作用于系统，即：

$$\boldsymbol{u}(t) = \boldsymbol{u}(t-1) + \Delta\boldsymbol{u}_t^* \tag{3.19}$$

系统执行这一控制量直到下一时刻。在新的时刻，系统根据状态信息重新

预测下一段时域的输出，通过优化过程得到一个新的控制增量序列。如此循环往复，直至系统完成控制过程。

3.3.2　非线性系统线性化方法

根据第2章车辆系统建模的知识，我们知道，无论是运动学建模，还是动力学建模，得到的都是一个非线性系统。显然，这种系统是不能直接用于线性时变模型预测控制的。本节介绍如何将一般的非线性系统进行线性化处理，以得到需要的线性时变系统。

将一个非线性系统近似为线性时变系统有很多方法，大体可以分为近似线性化与精确线性化两类。近似线性化方法简单，适用性较强，缺点是在控制精度要求非常高的场合难以适用；精确线性化一般不具备普遍性，往往是需要针对单个系统进行具体分析。在模型预测控制中，一般采用近似的线性化方法。以下介绍两种常用的线性化方法。

方式A：存在参考系统的线性化方法

该方法的主要思想是假设参考系统已经在期望路径上完全通过，得到了路径上每个时刻的状态量和控制量[50]。通过对参考系统和当前系统之间的偏差处理设计使模型预测控制器来跟踪期望路径。参考系统的任意时刻的状态和控制量满足如下关系：

$$\dot{\boldsymbol{\xi}}_r = f(\boldsymbol{\xi}_r, \boldsymbol{u}_r) \tag{3.20}$$

在任意点（$\boldsymbol{\xi}_r$、$\boldsymbol{u}_r$）处进行泰勒展开，只保留一阶项，忽略高阶项，得到：

$$\dot{\boldsymbol{\xi}} = f(\boldsymbol{\xi}_r, \boldsymbol{u}_r) + \left.\frac{\partial f}{\partial \boldsymbol{\xi}}\right|_{\substack{\boldsymbol{\xi}=\boldsymbol{\xi}_r \\ \boldsymbol{u}=\boldsymbol{u}_r}} (\boldsymbol{\xi} - \boldsymbol{\xi}_r) + \left.\frac{\partial f}{\partial \boldsymbol{u}}\right|_{\substack{\boldsymbol{\xi}=\boldsymbol{\xi}_r \\ \boldsymbol{u}=\boldsymbol{u}_r}} (\boldsymbol{u} - \boldsymbol{u}_r) \tag{3.21}$$

也可以写为：

$$\dot{\boldsymbol{\xi}} = f(\boldsymbol{\xi}_r, \boldsymbol{u}_r) + J_f(\boldsymbol{\xi})(\boldsymbol{\xi} - \boldsymbol{\xi}_r) + J_f(\boldsymbol{u})(\boldsymbol{u} - \boldsymbol{u}_r) \tag{3.22}$$

式中，$J_f(\boldsymbol{\xi})$为f相对于$\boldsymbol{\xi}$的雅可比矩阵，$J_f(\boldsymbol{u})$为f相对于$\boldsymbol{u}$的雅可比矩阵。

将式（3.22）与式（3.20）相减可以得到：

$$\dot{\tilde{\boldsymbol{\xi}}} = \boldsymbol{A}(t)\tilde{\boldsymbol{\xi}} + \boldsymbol{B}(t)\tilde{\boldsymbol{u}} \tag{3.23}$$

式中，$\tilde{\boldsymbol{\xi}} = \boldsymbol{\xi} - \boldsymbol{\xi}_r$，$\tilde{\boldsymbol{u}} = \boldsymbol{u} - \boldsymbol{u}_r$，$\boldsymbol{A}(t) = J_f(\boldsymbol{\xi})$，$\boldsymbol{B}(t) = J_f(\boldsymbol{u})$。

这样，就得到新的状态方程。该状态方程是连续的，不能直接用于模型预测控制器的设计，故有必要对其进行离散化处理。此处采用近似的离散化，即：

$$\begin{aligned} \boldsymbol{A}_{k,t} &= \boldsymbol{I} + T\boldsymbol{A}(t) \\ \boldsymbol{B}_{k,t} &= T\boldsymbol{B}(t) \end{aligned} \tag{3.24}$$

结合式（3.23）和式（3.24），可以得到：

$$\tilde{\boldsymbol{\xi}}(k+1) = \boldsymbol{A}_{k,t}\tilde{\boldsymbol{\xi}}(k) + \boldsymbol{B}_{k,t}\tilde{\boldsymbol{u}}(k) \tag{3.25}$$

至此，得到了非线性系统在任意一个参考点处线性化后的系统，该系统是设计线性模型预测控制算法的基础。

方式 B：针对状态轨迹的线性化方法

该方法的主要思想是通过对系统施加持续不变的控制量，得到一条状态轨迹，根据该状态轨迹和系统实际状态量的偏差设计线性模型预测控制算法。这种方法主要的优势是不需要预先得到期望跟踪路径的状态量和控制量。如果方式 A 中参考系统的控制量始终不变，则与此方法等同。

考虑系统的某个工作点为 $[\boldsymbol{\xi}_0, \boldsymbol{u}_0]$，$\dot{\boldsymbol{\xi}}_0(k)$ 为始终施加控制量为 $\boldsymbol{u}_0$ 后得到的系统状态量，则存在以下关系：

$$\begin{aligned} \dot{\boldsymbol{\xi}}_0(k+1) &= f(\dot{\boldsymbol{\xi}}_0(k), \boldsymbol{u}_0) \\ \dot{\boldsymbol{\xi}}_0(0) &= \boldsymbol{\xi}_0 \end{aligned}$$

经过和方式 A 类似的推导，可以得到：

$$\tilde{\boldsymbol{\xi}}(k+1) = \boldsymbol{A}_{k,t}\tilde{\boldsymbol{\xi}}(k) + \boldsymbol{B}_{k,t}\tilde{\boldsymbol{u}}(k)$$

式中，$\tilde{\boldsymbol{\xi}} = \boldsymbol{\xi}(k) - \dot{\boldsymbol{\xi}}_0(k)$，$\tilde{\boldsymbol{u}} = \boldsymbol{u}(k) - \boldsymbol{u}_0$。

针对该方法还有其他的数学处理方法，具体可参考文献［70］；同时，除了以上两种方法外，还有其他的一些线性化方法，例如分段线性化等，可参考文献［71］。

3.3.3 工程实例

上一节我们从理论上推导了线性时变模型预测控制，而本节将通过一个工程实例进一步加强读者对这一理论的认识，并给出相应的程序代码和分析。

（1）控制系统简介

以无人驾驶车辆的轨迹跟踪问题作为应用背景。轨迹跟踪是机器人运动控制的基本问题之一。无人驾驶车辆在一个给定位置出发，通过离散轨迹点或者连续轨迹函数的指引，最终跟踪上期望轨迹。详细的介绍可以阅读文献［50］。

轨迹跟踪仿真基本设定为：车辆从坐标原点出发，以期望纵向速度 $v = 1$ m/s 跟踪一条直线轨迹 $y = 2$。采样时间为 50 ms，仿真总时间设定为 20 s，仿真环境为 Matlab。

（2）问题分析

由于控制目标是无人驾驶车辆在低速情况下的跟踪控制，因此考虑以车辆运动学方程作为预测模型。根据第2章的介绍，低速情况下的车辆运动学方程形式如下：

$$\begin{bmatrix}\dot{x}_r\\ \dot{y}_r\\ \dot{\varphi}\end{bmatrix}=\begin{bmatrix}\cos\varphi\\ \sin\varphi\\ \tan\delta_f/l\end{bmatrix}v_r$$

对其进行线性化，得到线性时变模型为：

$$\tilde{\boldsymbol{\xi}}_{kin}(k+1)=\boldsymbol{A}_{kin}(k)\tilde{\boldsymbol{\xi}}_{kin}(k)+\boldsymbol{B}_{kin}(k)\tilde{\boldsymbol{u}}_{kin}(k) \tag{3.26}$$

式中，各矩阵和状态变量为：

$$\boldsymbol{\xi}_{kin}=\begin{bmatrix}x-x_r\\ y-y_r\\ \varphi-\varphi_r\end{bmatrix}\quad \boldsymbol{A}_{kin}(k)=\begin{bmatrix}1 & 0 & -v_r\sin\varphi_r T\\ 0 & 1 & v_r\cos\varphi_r T\\ 0 & 0 & 1\end{bmatrix},$$

$$\boldsymbol{B}_{kin}(k)=\begin{bmatrix}\cos\varphi_r T & 0\\ \sin\varphi_r T & 0\\ \dfrac{\tan\delta_{f,r}T}{l} & \dfrac{v_r T}{l\cos^2(\delta_{f,r})}\end{bmatrix},$$

T 为采样时间

（3）代码解析

在 Matlab 环境下对无人驾驶车辆的直线轨迹跟踪过程进行仿真。为了更清晰地分析整个程序，将仿真程序 chapter3 - 3 - 3.m 分为若干模块，分别介绍。第1部分是参考轨迹点生成的程序：

```
% chapter3 - 3 - 3.m 第 1 部分
% 参考轨迹生成
N = 100;                    % 参考轨迹点数量
T = 0.05;                   % 采样周期
Xout = zeros(N,3);
Tout = zeros(N,1);
for k = 1:1:N
    Xout(k,1) = k * T;
    Xout(k,2) = 2;
```

```
    Xout(k,3)=0;
    Tout(k,1)=(k-1)*T;
end
```

参考轨迹点生成是根据仿真设定给出的，读者可以尝试自己编写其他形式的参考轨迹，如圆形轨迹、8字形轨迹等。

程序 chapter3-3-3.m 第2部分主要描述控制系统的基本情况，包括状态量和控制量个数、初始状态等：

```
% chapter3-3-3.m 第2部分
% 仿真系统基本情况介绍
Nx=3;                % 状态量个数
Nu=2;                % 控制量个数
[Nr,Nc]=size(Xout);
Tsim=20;仿真时间
X0=[0 0 pi/3];车辆初始状态
L=1;                 % 车辆轴距
vd1=1;               % 参考系统的纵向速度
vd2=0;               % 参考系统的前轮偏角
```

程序 chpater3-3-3.m 第3部分是根据控制系统的维度信息，提前定义好相关矩阵并赋值。考虑到读者大多数都是第一次编写模型预测控制的应用程序，此处以文献［70］中的方法为基础编写，对于介绍的第2种目标函数及其解法，将在第4章中采用。

```
% chapter3-3-3.m 第3部分
% 矩阵定义
x_real=zeros(Nr,Nc);
x_piao=zeros(Nr,Nc);
u_real=zeros(Nr,Nu);
u_piao=zeros(Nr,Nu);
x_real(1,:)=X0;
x_piao(1,:)=x_real(1,:)-Xout(1,:);
X_PIAO=zeros(Nr,Nx*Tsim);
XXX=zeros(Nr,Nx*Tsim);% 用于保存每个时刻预测的所有状态值
```

```
q=[1 0 0;0 1 0;0 0 0.5];
Q_cell=cell(Tsim,Tsim);
for i=1:1:Tsim
    for j=1:1:Tsim
        if i==j
            Q_cell{i,j}=q;
        else
            Q_cell{i,j}=zeros(Nx,Nx);
        end
    end
end
Q=cell2mat(Q_cell);
R=0.1*eye(Nu*Tsim,Nu*Tsim);
```

程序 chapter3 -3 -3. m 第 4 部分是模型预测控制的主体。在每一控制周期的开始，获取系统当前状态量，更新状态空间方程。根据 3.3.1 节中的方法，求解一个标准二次规划问题。将所求序列的第一个解施加到系统上。由于本仿真程序只运行在 Matlab 环境下，因此车辆的运动过程以运动微分方程求解的形式产生。为了直观体验模型预测控制在车辆跟踪过程中的控制效果，加入了绘图程序，这样能够动态显示车辆运行结果。


```
% chapter3 -3 -3.m 第4 部分
for i=1:1:Nr
    t_d=Xout(i,3);
    a=[1    0    -vd1*sin(t_d)*T;
       0    1     vd1*cos(t_d)*T;
       0    0     1;];
    b=[cos(t_d)*T   0;
       sin(t_d)*T   0;
       vd2*T/L      vd1*T/(cos(vd2)^2);];
A_cell=cell(Tsim,1);
B_cell=cell(Tsim,Tsim);
for j=1:1:Tsim
    A_cell{j,1}=a^j;
```

```
    for k=1:1:Tsim
        if k<=j
                B_cell{j,k}=(a^(j-k))*b;
        else
                B_cell{j,k}=zeros(Nx,Nu);
        end
      end
    end
    A=cell2mat(A_cell);
    B=cell2mat(B_cell);

    H=2*(B'*Q*B+R);
    f=2*B'*Q*A*x_piao(i,:)';
    A_cons=[];
    b_cons=[];
    lb=[-2.2;-0.64];
    ub=[0.2;0.64];
    [X,fval(i,1),exitflag(i,1),output(i,1)]=quadprog(H,f,
A_cons,b_cons,[],[],lb,ub);
    X_PIAO(i,:)=(A*x_piao(i,:)'+B*X)';
    if i+j<Nr
          for j=1:1:Tsim
          XXX(i,1+3*(j-1))=X_PIAO(i,1+3*(j-1))+Xout(i+j,1);
          XXX(i,2+3*(j-1))=X_PIAO(i,2+3*(j-1))+Xout(i+j,2);
          XXX(i,3+3*(j-1))=X_PIAO(i,3+3*(j-1))+Xout(i+j,3);
          end
    else
          for j=1:1:Tsim
          XXX(i,1+3*(j-1))=X_PIAO(i,1+3*(j-1))+Xout(Nr,1);
          XXX(i,2+3*(j-1))=X_PIAO(i,2+3*(j-1))+Xout(Nr,2);
          XXX(i,3+3*(j-1))=X_PIAO(i,3+3*(j-1))+Xout(Nr,3);
      end
    end
    u_piao(i,1)=X(1,1);
```

```
u_piao(i,2)=X(2,1);
Tvec=[0:0.05:4];
X00=x_real(i,:);
vd11=vd1+u_piao(i,1);
vd22=vd2+u_piao(i,2);
XOUT=dsolve('Dx-vd11*cos(z)=0','Dy-vd11*sin(z)=0',
'Dz-vd22=0','x(0)=X00(1)','y(0)=X00(2)','z(0)=X00(3)');
t=T;
x_real(i+1,1)=eval(XOUT.x);
x_real(i+1,2)=eval(XOUT.y);
x_real(i+1,3)=eval(XOUT.z);
if(i<Nr)
     x_piao(i+1,:)=x_real(i+1,:)-Xout(i+1,:);
end
u_real(i,1)=vd1+u_piao(i,1);
u_real(i,2)=vd2+u_piao(i,2);

figure(1);
plot(Xout(1:Nr,1),Xout(1:Nr,2));
hold on;
plot(x_real(i,1),x_real(i,2),'r*');
xlabel('X[m]');
axis([-15 -13]);
ylabel('Y[m]');
hold on;
      for k=1:1:Tsim
            X(i,k+1)=XXX(i,1+3*(k-1));
            Y(i,k+1)=XXX(i,2+3*(k-1));
      end
      X(i,1)=x_real(i,1);
      Y(i,1)=x_real(i,2);
      plot(X(i,:),Y(i,:),'y')
      hold on;
```

```
end
% 程序结束
```

结合以上4个部分，在Matlab环境下运行可以得到结果，如图3.11所示。图3.11中的曲线a为期望轨迹，星号曲线b为车辆位置，曲线c为预测时域内车辆位置。从图3.11可以看出，车辆在模型预测控制器的作用下快速跟踪上了期望轨迹，最后能够沿着期望轨迹稳定行驶，达到了预期效果；同时，我们通过这张图也能直观看出模型预测控制中滚动优化的含义，即在每一采样周期都会预测系统在一段时域的状态信息，进而通过优化求解得到一个控制序列。另外，我们也应当注意到一个细节，就是车辆实际经过的位置与预测的位置并不重合。这主要是因为采用了线性误差模型作为预测模型，而车辆位置的解算采用的是非线性运动学模型。这种现象在实际应用中普遍存在，因为预测模型往往不是对被控系统的精确建模，而是对系统的合理简化。

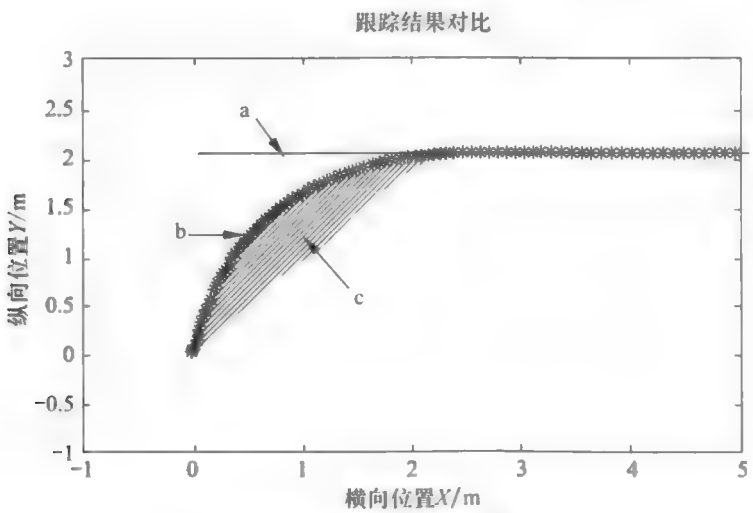


图3.11 实例运行结果

为了进一步了解系统运行过程中其他状态量的情况，可以通过以下绘图程序进行：

```
% 系统状态量随时间变化
figure(2)
subplot(3,1,1);
plot(Tout(1:Nr),Xout(1:Nr,1));
```

```
hold on;
plot(Tout(1:Nr),x_real(1:Nr,1),'r');
xlabel('采样时间 T');
ylabel('横向位置 X')
subplot(3,1,2);
plot(Tout(1:Nr),Xout(1:Nr,2));
hold on;
plot(Tout(1:Nr),x_real(1:Nr,2),'r');
xlabel('采样时间 T');
ylabel('纵向位置 Y')
subplot(3,1,3);
plot(Tout(1:Nr),Xout(1:Nr,3));
hold on;
plot(Tout(1:Nr),x_real(1:Nr,3),'r');
hold on;
xlabel('采样时间 T');
ylabel('\theta')
% ========================================================
% 控制量随时间变化
% ========================================================
figure(3)
subplot(2,1,1);
plot(Tout(1:Nr),u_real(1:Nr,1),'r');
xlabel('采样时间 T');
ylabel('纵向速度')
subplot(2,1,2)
plot(Tout(1:Nr),u_real(1:Nr,2),'r');
xlabel('采样时间 T');
ylabel('角速度')
% ========================================================
% 状态量偏差随时间变化
% ========================================================
figure(4)
subplot(3,1,1);
```

```
plot(Tout(1:Nr),x_piao(1:Nr,1));
xlabel('采样时间 T');
ylabel('e(x)');
subplot(3,1,2);
plot(Tout(1:Nr),x_piao(1:Nr,2));
xlabel('采样时间 T');
ylabel('e(y)');
subplot(3,1,3);
plot(Tout(1:Nr),x_piao(1:Nr,3));
xlabel('采样时间 T');
ylabel('e(\theta)');
% 程序结束
```

绘图结果如图 3.12、图 3.13 和图 3.14 所示。其中图 3.12 中虚线为参考状态量，而实线为跟踪过程中实际的控制量。从图中可以看出，无人驾驶车辆在存在初始误差的情况下能快速消除误差并最终达到稳定状态。

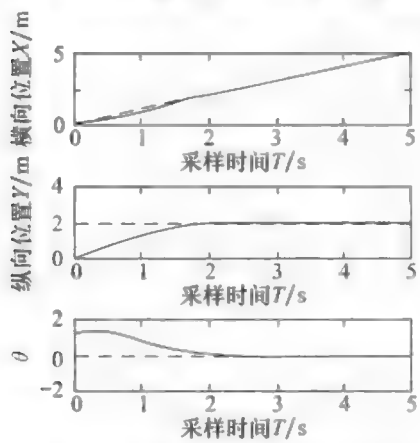


图 3.12　系统状态量随时间变化

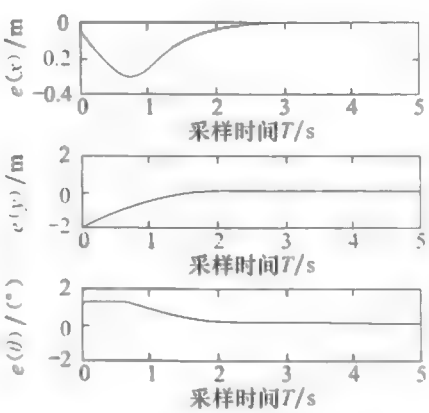


图 3.13　状态量偏差随时间变化

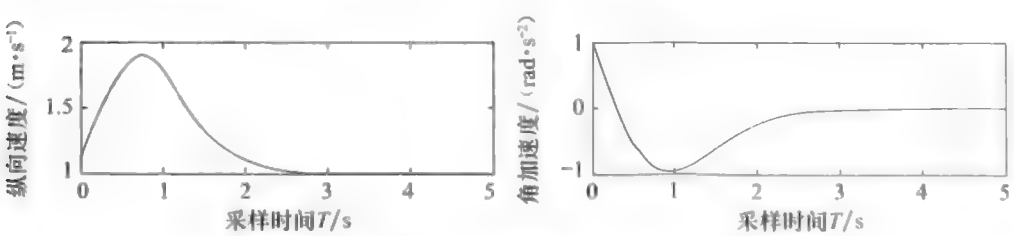


图 3.14　控制量随时间变化

3.4　非线性模型预测控制算法

3.4.1　问题描述

在上面的实例中，为了应用线性时变模型预测控制，对一个非线性车辆运动学模型进行了线性化．相对于非线性模型预测控制，线性时变模型预测控制是一种次优的选择．随着研究的深入与硬件水平的提高，非线性模型预测控制的应用范围也愈加广阔．

对于一个非线性系统，考虑如下一般形式的离散模型：

$$\begin{aligned}\boldsymbol{\xi}(t+1) &= f(\boldsymbol{\xi}(t),\boldsymbol{u}(t))\\ \boldsymbol{\xi}(t) &\in \boldsymbol{\mathcal{X}},\quad \boldsymbol{u}(t) \in \boldsymbol{\Gamma}\end{aligned} \tag{3.27}$$

式中，$f(\cdot,\cdot)$ 为系统的状态转移函数，$\boldsymbol{\xi}$ 为 n_ξ 维状态变量，$\boldsymbol{u}$ 为 m_u 维控制变量，$\boldsymbol{\mathcal{X}}$ 为状态变量约束，$\boldsymbol{\Gamma}$ 为控制变量约束．

设定 $f(\mathbf{0},\mathbf{0})=\mathbf{0}$ 为系统的一个稳定点，同时也是系统的控制目标．对于任意的时域 N，考虑如下的优化目标函数 $J_N(\cdot,\cdot)$：

$$J_N(\boldsymbol{\xi}(t),\boldsymbol{U}(t)) = \sum_{k=t}^{t+N-1} l(\boldsymbol{\xi}(k),\ \boldsymbol{u}(k)) + P(\boldsymbol{\xi}(t+N)) \tag{3.28}$$

式中，$\boldsymbol{U}(t)=[\boldsymbol{u}(t),\cdots,\boldsymbol{u}(t+N-1)]^{\mathrm{T}}$ 是在时域 N 内的控制量输入序列，$\boldsymbol{\xi}(t)$ 是在输入向量序列 $\boldsymbol{U}(t)$ 作用于系统下的状态向量轨迹．优化目标函数中的第一项 $l(\cdot,\cdot)$ 表征对期望输出的跟踪能力，第二项 $P(\cdot)$ 表征终端约束．

结合式（3.27）和式（3.28），非线性模型预测控制就是要在每一个步长内求解以下带约束的有限时域优化问题：

$$\min_{\boldsymbol{U}_t,\boldsymbol{\xi}_{t+1,t},\cdots,\boldsymbol{\xi}_{t+N,t}} J_N(\boldsymbol{\xi}_t,\boldsymbol{U}_t) \tag{3.29a}$$

$$s.t.\quad \boldsymbol{\xi}_{k+1,t} = f(\boldsymbol{\xi}_{k,t},\boldsymbol{u}_{k,t}),\quad k=t,\cdots,N-1 \tag{3.29b}$$

$$\boldsymbol{\xi}_{k,t} \in \boldsymbol{\mathcal{X}}\quad k=t+1,\cdots,t+N-1 \tag{3.29c}$$

$$\boldsymbol{u}_{k,t} \in \boldsymbol{\Gamma}\quad k=t,\cdots,t+N-1 \tag{3.29d}$$

$$\boldsymbol{\xi}_{t,t} = \boldsymbol{\xi}(t) \tag{3.29e}$$

$$\boldsymbol{\xi}_{N,t} \in \boldsymbol{\mathcal{X}}_{\mathrm{fin}} \tag{3.29f}$$

其中，式（3.29b）为系统所决定的状态约束，式（3.29c）和式（3.29d）分别为状态向量和控制输入向量约束，式（3.29e）为初始状态约束，式（3.29f）为终端状态约束．

假定上述优化问题存在可行解，通过求解该问题，可以得到最优控制序列 $\boldsymbol{U}_t^*(t)=[\boldsymbol{u}_{t,t}^*, \cdots, \boldsymbol{u}_{t+N-1,t}^*]$。根据模型预测控制的原理，只是将该控制序列中的第一个元素作为受控对象的实际控制输入，即：

$$\boldsymbol{u}(\boldsymbol{\xi}(t)) = \boldsymbol{u}_{t,t}^* \tag{3.30}$$

在下一个采样时刻，系统重新以新的采样时刻为初始状态求解式（3.29），并继续将控制序列的第一个元素施加给受控对象，如此循环，直至完成整个控制过程。对于任意的非线性模型预测问题，在方程求解的整个过程中包含了 $N(n+m)$ 个最优变量，nN 个非线性状态约束；同时，也包含了由控制输入约束和状态向量约束决定的一系列线性约束。因此，对于非线性模型预测控制，其求解难度会随着系统状态维度的增加而迅速增加。对于阶次较低的非线性系统，如车辆3自由度运动学模型，非线性模型预测控制的优势依然存在；而对于阶次较高的非线性系统，如车辆多自由度动力学模型，不进行适当的简化，就很难实现在线实时控制。

3.4.2 非线性模型预测控制的数值解法

在非线性模型预测控制中，通过非线性模型、当前的状态量和控制时域内的控制量序列对未来的状态量进行预测。显然，这是一个迭代的过程，由于控制序列不可知，就需要找到一个显式的迭代方程对微分方程进行近似求解。在工程实际中，使用较为广泛的数值解法有欧拉法（Euler Method）和4阶龙格－库塔（Runge－Kutta）算法。

对于微分方程 $\dot{y}=f(t, y)$，利用单步的欧拉法，可以得到 $n+1$ 时刻相对于 n 时刻的迭代方程为：

$$y_{n+1} = y_n + hf(t_n, y_n) \tag{3.31}$$

式中，h 为迭代步长。

单步的欧拉法形式简单，但精度较低。当应用场合需要较高的精度时，通常采用4阶龙格－库塔算法。其一般表述形式为：

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{3.32}$$

式中，$k_1=f(t_n, y_n)$，$k_2=f\left(t_n+\frac{h}{2}, y_n+\frac{h}{2}k_1\right)$，$k_3=f\left(t_n+\frac{h}{2}, y_n+\frac{h}{2}k_2\right)$，$k_4=f(t_n+h, y_n+hk_3)$。

4阶龙格－库塔算法每步的误差为 h^5 阶，总积累误差为 h^4 阶。迭代步长越小，计算精度越高，但计算速度也会下降。

3.4.3　工程实例

本节依然以无人驾驶车辆的轨迹跟踪问题作为应用背景。读者在阅读过程中可以把两种方法进行对比，这样就可以更加深入理解两种算法的区别。

首先我们单独定义一个 m 函数（chapter3_4_3.m），用于计算非线性模型预测控制的目标函数，命名为 MY_costfunction。在这个函数里，我们需要根据车辆的当前状态信息和未来的控制序列去预测车辆的未来状态信息，并计算目标函数值。传递参数设置为待求状态量、当前状态量、预测时域、控制时域、采样周期、参考轨迹和权重矩阵，包含了我们设计非线性模型预测控制器所必需的参数。程序如下：

```
function cost = MY_costfunction(x,State_Initial,Np,Nc,T,
Xref,Yref,PHIref,Q,R)
    cost = 0;                          % 初始化目标函数值
    l = 1;                             % 车辆轴距
    % ------------------------------------------------------
    % 矩阵初始化，包含当前状态、预测状态、状态偏差和控制量
    % ------------------------------------------------------
    X = State_Initial(1,1);
    Y = State_Initial(2,1);
    PHI = State_Initial(3,1);

    X_predict = zeros(Np,1);
    Y_predict = zeros(Np,1);
    PHI_predict = zeros(Np,1);

    X_error = zeros(Np+1,1);
    Y_error = zeros(Np+1,1);
    PHI_error = zeros(Np+1,1);

    v = zeros(Np,1);                   % 控制量 v
    delta_f = zeros(Np,1);             % 控制量 delta_f
    % ------------------------------------------------------
```

```
% 状态更新
% ==========================================================
for i =1:1:Np
    if i ==1
        v(i,1)=x(1);
        delta_f(i,1)=x(2);
        X_predict(i,1)=X+T*v(i,1)*cos(PHI);
        Y_predict(i,1)=Y+T*v(i,1)*sin(PHI);
        PHI_predict(i,1)=PHI+T*v(i,1)*tan(delta_
        f(i,1))/l;
    else
        v(i,1)=x(3);
        delta_f(i,1)=x(4);
        X_predict(i,1)=X_predict(i-1)+T*v(i,1)*cos
        (PHI_predict(i-1));
        Y_predict(i,1)=Y_predict(i-1)+T*v(i,1)*sin
        (PHI_predict(i-1));
        PHI_predict(i,1)=PHI_predict(i-1)+T*v(i,1)
        *tan(delta_f(i,1))/l;
end
% ==========================================================
% 计算预测时域内的轨迹偏差
% ==========================================================
    X_real=zeros(Np+1,1);
    Y_real=zeros(Np+1,1);
    X_real(1,1)=X;
    X_real(2:Np+1,1)=X_predict;
    Y_real(1,1)=Y;
    Y_real(2:Np+1,1)=Y_predict;
    X_error(i,1)=X_real(i,1)-Xref(i,1);
    Y_error(i,1)=Y_real(i,1)-Yref(i,1);
    PHI_error(i,1)=PHI_predict(i,1)-PHIref(i,1);
end
% ==========================================================
```

```
% 计算目标函数值
% ================================================================
cost = cost + Y_error' * R * Y_error + X_error' * Q * X_error;
% 程序结束
```

有了上面程序的基础，对于非线性模型预测控制方法的处理就会简单很多，关键步骤仍然和3.3.3节中介绍的一样，首先给定系统的基本参数，然后生成参考轨迹，最后对目标函数进行求解。为了动态表示无人驾驶车辆对直线轨迹的跟踪过程，就需要在求解之后实时绘制车辆的位置，程序（chapter3_4_3.m）如下：

```
clc;
clear all;
% ================================================================
% 参数初始化
% ================================================================
Nx = 3;                          % 状态量个数
Np = 15;                         % 预测时域
Nc = 2;                          % 控制时域
l = 1;                           % 车辆轴距

State_Initial = zeros(Nx,1);     % 状态矩阵初始化
State_Initial(1,1) = 0;          % x
State_Initial(2,1) = 0;          % y
State_Initial(3,1) = pi/6;       % phi

Q = 100 * eye(Np+1,Np+1);        % 权重矩阵
R = 100 * eye(Np+1,Np+1);        % 权重矩阵

% ----------------
% 参考轨迹生成
% ----------------
N = 100;                         % 参考轨迹点数量
T = 0.05;                        % 采样周期
Xref = zeros(Np,1);
```

```
Yref = zeros(Np,1);
PHIref = zeros(Np,1);
for Nref = 1:1:Np
            Xref(Nref,1)=(j+Nref-1)*T;
            Yref(Nref,1)=2;
            PHIref(Nref,1)=0;
end
% ================================================================
% 开始进行求解
% ================================================================
for j = 1:1:N
      lb=[0.8;-0.44;0.8;-0.44];
      ub=[1.2;0.44;1.2;0.44];
      A=[];
      b=[];
      Aeq=[];
      beq=[];
      options = optimset('Algorithm','active-set');
      [A,fval,exitflag]= fmincon(@(x)MY_costfunction(x,
      State_Initial,Np,Nc,T,Xref,Yref,PHIref,Q,R),[0;0;
      0;0;],A,b,Aeq,beq,lb,ub,[],options);    % 有约束求解
      v_actual = A(1);
      deltaf_actual = A(2);

      X00(1)= State_Initial(1,1);
      X00(2)= State_Initial(2,1);
      X00(3)= State_Initial(3,1);
      XOUT = dsolve('Dx-v_actual*cos(z)=0','Dy-v_actual
      *sin(z)=0','Dz-v_actual*tan(deltaf_actual)=0','x
      (0)=X00(1)','y(0)=X00(2)','z(0)=X00(3)');
      t = T;
      State_Initial(1,1)= eval(XOUT.x);
      State_Initial(2,1)= eval(XOUT.y);
      State_Initial(3,1)= eval(XOUT.z);
```

```
    figure(1)
    plot(State_Initial(1,1),State_Initial(2,1),'b*');
    hold on;
    plot([0,5],[2,2],'r--');
    hold on;
    axis([0 5 0 4])
end
% 程序结束
```

运行之后的结果如图 3.15 所示。从图 3.15 中可以看出，无人驾驶车辆在控制器作用下成功跟踪上了期望轨迹，并且最后保持稳定，但在局部区域出现了超调的现象，而通过调整控制器的预测时域可以消除超调。图 3.16 是将预测

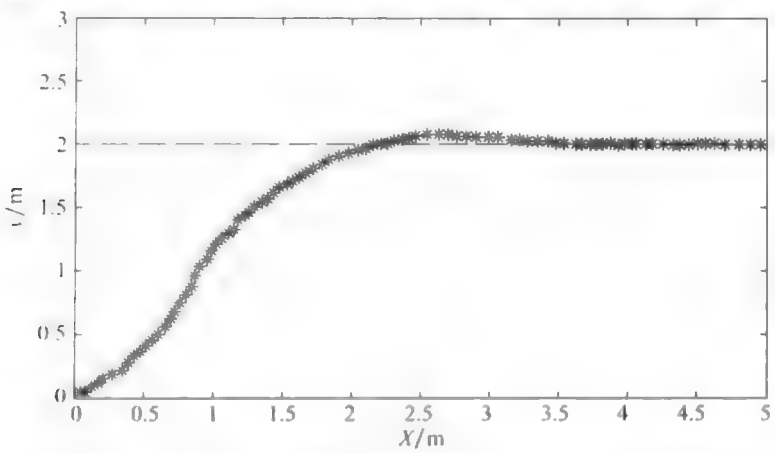


图 3.15　基于非线性模型预测控制的直线轨迹跟踪结果

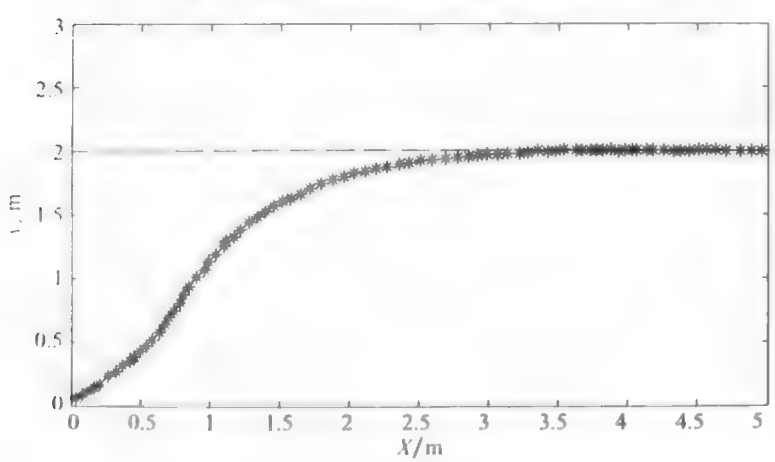


图 3.16　改变预测时域后的跟踪结果

时域由 15 增加到 25 时的结果，可以清楚地看到，已经没有了超调的现象。读者在编写程序的时候，也可以多尝试改变控制器参数，观看结果，这样可以对模型预测控制有更加深入的了解。

3.5 线性约束下的二次型规划控制算法

线性二次型调节器（Linear Quadratic Regulator，LQR）的研究对象是现代控制理论中以状态空间形式给出的线性系统，而目标函数为对象状态和（或）控制输入的二次型函数。LQR 的任务在于当系统状态由于任何原因偏离了平衡状态时，能在不消耗过多能量的情况下，保持系统状态各个分量仍接近于平衡状态。LQR 理论是现代控制理论中发展最早，也最为成熟的一种状态空间设计法。特别可贵的是，LQR 解出的控制规律是状态变量的线性函数，因此可以通过状态反馈实现闭环最优控制。这在实际工程应用中具有重要的意义。

3.5.1 线性约束转化为 LQR 问题

对于优化过程中的线性约束，可以通过拉格朗日乘子法求解。拉格朗日乘子法是一种寻找变量受一个或多个条件限制的多元函数的极值的方法。这种方法引入了一种新的标量未知数，即拉格朗日乘数。具体实现方法如下：

考虑系统：

$$\dot{\boldsymbol{x}}(t) = f[\boldsymbol{x}(t), \boldsymbol{u}(t), t] \tag{3.33}$$

式中，$\boldsymbol{x}(t)$ 为状态变量，$\boldsymbol{u}(t)$ 为控制量，$f[\boldsymbol{x}(t), \boldsymbol{u}(t), t]$ 为连续可微的矢量函数。

设给定 $t \in [t_0, t_f]$，初始状态 $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$，终端状态 $\boldsymbol{x}(t_f)$ 自由。二次型评价函数的一般形式如下：

$$J = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}(t_f)\boldsymbol{Q}_0(t)\boldsymbol{x}(t_f) + \frac{1}{2}\int_{t_0}^{t_f}[\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}(t)\boldsymbol{x} + \boldsymbol{u}^{\mathrm{T}}\boldsymbol{R}(t)\boldsymbol{u}]\mathrm{d}t \tag{3.34}$$

式中，$\boldsymbol{Q}(t)$ 为半正定的状态加权矩阵，$\boldsymbol{R}(t)$ 为正定的控制加权矩阵，$\boldsymbol{Q}_0(t)$ 为半正定的终端加权矩阵。在实际工程应用中 $\boldsymbol{Q}(t)$ 和 $\boldsymbol{R}(t)$ 是对称矩阵，常取对角阵。

式（3.34）中的第一项 $\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}(t_f)\boldsymbol{Q}_0(t)\boldsymbol{x}(t_f)$ 表出了对终端误差的要求，叫作终端代价函数。当对终端状态要求比较严格时，需要加入这一项，以体现在 t_f 时刻的误差是够小。被积函数中第一项 $L_x = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}(t)\boldsymbol{x}$，若 $\boldsymbol{x}$ 表示误差矢量，那么 L_x 表示跟踪过程中误差大小的代价函数。被积函数的第二项

$L_u = \frac{1}{2}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{R}(t)\boldsymbol{u}$，表示动态过程中对控制的约束或要求。$L_u$ 是用来衡量控制功率大小的代价函数。

通过使目标评价函数 J 取得最小值，寻求最优控制 $\boldsymbol{u}(t)$，将系统从初始状态转移到终端状态，实现以不大的控制保持较小的跟踪误差，达到能量和跟踪误差综合最优的目的。

为了求解线性约束下评价函数 J 的极值，应用拉格朗日乘子法构造增广泛函。首先将状态方程写成约束方程的形式：

$$f[\boldsymbol{x}(t),\boldsymbol{u}(t),t] - \dot{\boldsymbol{x}}(t) = 0 \tag{3.35}$$

然后通过引入待定的拉格朗日乘子矢量 $\boldsymbol{\lambda}(t)$，构造增广函数：

$$J = \int_{t_0}^{t_f}\{J + \boldsymbol{\lambda}^{\mathrm{T}}[f[\boldsymbol{x}(t),\boldsymbol{u}(t),t] - \dot{\boldsymbol{x}}(t)]\}\mathrm{d}t \tag{3.36}$$

至此，我们将一个含有线性约束的最优化问题转换为一个可以通过经典变分方法求解的方程组（其变量不受任何约束）的极值问题。

3.5.2　LQR 在无人驾驶车辆路径跟踪中的应用

在无人驾驶车辆路径跟踪的应用中，一般利用 LQR 构建前馈 + 反馈的控制系统：利用参考路径及其曲率获取前馈控制量，并采用 LQR 消除无人驾驶车辆当前状态与参考轨迹之间的状态误差。

给出由点序列 $S_0 = \{S_0, S_1, \cdots, S_N\}$ 组成的参考轨迹，如图 3.17 所示。可以根据参考轨迹的曲率获得前馈控制量 $u_0 = \{u_0, u_1, \cdots, u_N\}$，其中 u_k（$k = 1, 2, \cdots, N$）表示从点 S_{k-1} 到 S_k 的前馈控制量。

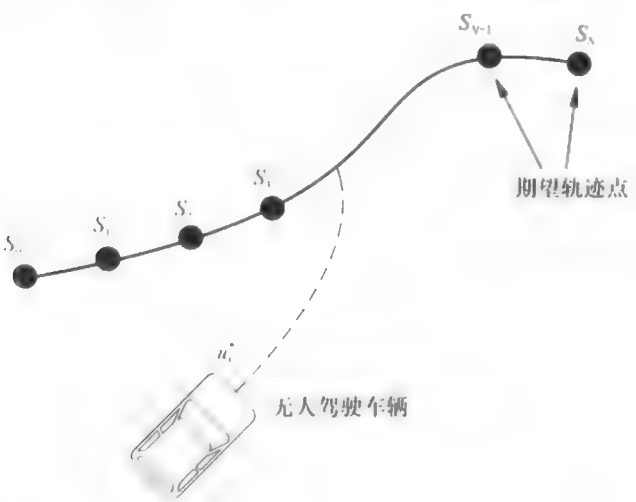


图 3.17　由参考路径获得前馈控制量

这里引用第2章的车辆运动学模型式（2.7）。由于这是一个非线性模型，故需要对其进行线性化，线性化过程请参考本书第4.2.1节，得到线性化的车辆运动状态空间方程如式（3.37）。由于LQR无法处理多变量控制过程中的约束问题，故这里不考虑无人驾驶车辆行驶过程中的约束因素

$$\Delta \boldsymbol{x}_{k+1} = \boldsymbol{A}_k \Delta \boldsymbol{x}_k + \boldsymbol{B}_k \Delta \boldsymbol{u}_k \tag{3.37}$$

式中，$\Delta \boldsymbol{x}_k$ 表示车辆当前的状态量与参考点的状态量的偏差，$\boldsymbol{A}_k = \boldsymbol{J}_x f(\boldsymbol{x}_k, \boldsymbol{u}_k)$，$\boldsymbol{B}_k = \boldsymbol{J}_u f(\boldsymbol{x}_k, \boldsymbol{u}_k)$，$\boldsymbol{J}_x$ 和 $\boldsymbol{J}_u$ 为函数 f 关于 x 和 u 的Jacobian矩阵。

使用LQR消除车辆当前状态与参考路径上参考点的状态误差，首先定义如下的评价函数：

$$J = \frac{1}{2}\sum_{k=0}^{N-1}\left[(\boldsymbol{x}_k - \boldsymbol{r}_k)^{\mathrm{T}}\boldsymbol{Q}(\boldsymbol{x}_k - \boldsymbol{r}_k) + (\boldsymbol{u}_k + \Delta \boldsymbol{u}_k)^{\mathrm{T}}\boldsymbol{R}(\boldsymbol{u}_k + \Delta \boldsymbol{u}_k)\right] + \frac{1}{2}(\boldsymbol{x}_N - \boldsymbol{r}_N)^{\mathrm{T}}\boldsymbol{Q}_0(t)(\boldsymbol{x}_N - \boldsymbol{r}_N) \tag{3.38}$$

式中，$\boldsymbol{Q}$ 为状态量误差的权重矩阵，$\boldsymbol{R}$ 为控制量的权重矩阵，$\boldsymbol{Q}_0$ 为终端状态权重矩阵，$\boldsymbol{x}_k$ 为无人驾驶车辆在第 k 时刻的预测状态，$\boldsymbol{r}_k$ 为参考路径上第 k 个参考点的状态，$\boldsymbol{u}_k + \Delta \boldsymbol{u}_k$ 为第 k 时刻的控制量。

引入拉格朗日乘子，构造无约束的最优化问题，如下：

$$V = \frac{1}{2}(\boldsymbol{x}_N - \boldsymbol{r}_N)^{\mathrm{T}}\boldsymbol{Q}_0(t)(\boldsymbol{x}_N - \boldsymbol{r}_N) + \frac{1}{2}\sum_{k=0}^{N-1}\left[(\boldsymbol{x}_k - \boldsymbol{r}_k)^{\mathrm{T}}\boldsymbol{Q}(\boldsymbol{x}_k - \boldsymbol{r}_k) + (\boldsymbol{u}_k + \Delta \boldsymbol{u}_k)^{\mathrm{T}}\boldsymbol{R}(\boldsymbol{u}_k + \Delta \boldsymbol{u}_k) + \boldsymbol{\lambda}_{k+1}^{\mathrm{T}}(\boldsymbol{A}_k \Delta \boldsymbol{x}_k + \boldsymbol{B}_k \Delta \boldsymbol{u}_k - \Delta \boldsymbol{x}_{k+1})\right] \tag{3.39}$$

构造哈密尔顿函数，令

$$H_k = \frac{1}{2}\left[(\boldsymbol{x}_k - \boldsymbol{r}_k)^{\mathrm{T}}\boldsymbol{Q}(\boldsymbol{x}_k - \boldsymbol{r}_k) + (\boldsymbol{u}_k + \Delta \boldsymbol{u}_k)^{\mathrm{T}}\boldsymbol{R}(\boldsymbol{u}_k + \Delta \boldsymbol{u}_k) + \boldsymbol{\lambda}_{k+1}^{\mathrm{T}}(\boldsymbol{A}_k \Delta \boldsymbol{x}_k + \boldsymbol{B}_k \Delta \boldsymbol{u}_k)\right] \tag{3.40}$$

将式（3.40）代入式（3.39），可得：

$$V = \frac{1}{2}(\boldsymbol{x}_N - \boldsymbol{r}_N)^{\mathrm{T}}\boldsymbol{Q}_0(t)(\boldsymbol{x}_N - \boldsymbol{r}_N) + \frac{1}{2}\left[H_0 + \boldsymbol{\lambda}_1^{\mathrm{T}}\Delta \boldsymbol{x}_1 + \sum_{k=1}^{N-1}(H_k - \boldsymbol{\lambda}_k^{\mathrm{T}}\Delta \boldsymbol{x}_k)\right] \tag{3.41}$$

当评价函数取最小值时，函数 V 需满足：

$$\begin{cases} \dfrac{\partial V}{\partial(\Delta \boldsymbol{x}_k)} = 0 \\ \dfrac{\partial V}{\partial(\Delta \boldsymbol{u}_k)} = 0 \\ \dfrac{\partial V}{\partial(\Delta \boldsymbol{x}_N)} = 0 \end{cases} \tag{3.42}$$

由式（3.42）可得：

$$\boldsymbol{Q}\Delta \boldsymbol{x}_k + \boldsymbol{A}_k \boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k \tag{3.43}$$

$$\boldsymbol{R}(\boldsymbol{u}_k + \Delta \boldsymbol{u}_k) + \boldsymbol{B}_k^{\mathrm{T}} \boldsymbol{\lambda}_{k+1} = 0 \tag{3.44}$$

$$\boldsymbol{Q}_0 \Delta \boldsymbol{x}_N = \boldsymbol{\lambda}_N \tag{3.45}$$

根据式（3.43），假设：

$$\boldsymbol{\lambda}_k = \boldsymbol{S}_k \Delta \boldsymbol{x}_k + \boldsymbol{v}_k \tag{3.46}$$

由式（3.38），式（3.43～3.45）和式（3.46）可得：

$$\begin{aligned} \boldsymbol{K} &= (\boldsymbol{B}_k^{\mathrm{T}} \boldsymbol{S}_{k+1} \boldsymbol{B}_k + \boldsymbol{R})^{-1} \boldsymbol{B}_k^{\mathrm{T}} \boldsymbol{S}_{k+1} \boldsymbol{A}_k \\ \boldsymbol{K}_u &= (\boldsymbol{B}_k^{\mathrm{T}} \boldsymbol{S}_{k+1} \boldsymbol{B}_k + \boldsymbol{R})^{-1} \boldsymbol{R} \\ \boldsymbol{K}_v &= (\boldsymbol{B}_k^{\mathrm{T}} \boldsymbol{S}_{k+1} \boldsymbol{B}_k + \boldsymbol{R})^{-1} \boldsymbol{B}_k^{\mathrm{T}} \\ \boldsymbol{S}_k &= \boldsymbol{A}_k^{\mathrm{T}} \boldsymbol{S}_{k+1} (\boldsymbol{A}_k - \boldsymbol{B}_k \boldsymbol{K}) + \boldsymbol{Q} \\ \boldsymbol{v}_k &= (\boldsymbol{A}_k - \boldsymbol{B}_k \boldsymbol{K}) \boldsymbol{v}_{k+1} - \boldsymbol{K}^{\mathrm{T}} \boldsymbol{R} \boldsymbol{u}_k \end{aligned} \tag{3.47}$$

根据已知的 $\boldsymbol{S}_N = \boldsymbol{Q}_0$，$\boldsymbol{v}_N = 0$ 和终端状态 $\boldsymbol{x}_N$，通过向后迭代可以求得反馈控制的最优控制序列，如下式所示：

$$\Delta \boldsymbol{u}_k = -\boldsymbol{K}\Delta \boldsymbol{x}_{k+1} - \boldsymbol{K}_u \boldsymbol{u}_k - \boldsymbol{K}_v \boldsymbol{v}_{k+1} \tag{3.48}$$

最终得到无人驾驶车辆的控制量为：

$$\boldsymbol{u}^* = \boldsymbol{u}_k + \Delta \boldsymbol{u}_k \tag{3.49}$$

式中，$\boldsymbol{u}_k$ 为前馈控制量，$\Delta \boldsymbol{u}_k$ 为反馈控制量

3.5.3　LQR 进行路径跟踪的工程实例

在工程应用中，首先完成车辆的初始状态的设置和系统参数的初始化，如下所示：

```
Hp = 10;                    % 设置 finite - horizon
N_1 = 200;                  % 设置迭代次数
Nx = 3;                     % 状态变量参数的个数
Nu = 1;                     % 控制变量参数的个数
```

```
FWA = zeros(N_1,1);              % 前轮偏角
FWA(1,1)=0;                      % 初始状态的前轮偏角
x_real = zeros(Nx,N_1);          % 实际状态
x_real(:,1)=[22 0 pi/2];         % x0 = 车辆初始状态 X_init 初始状
                                  态,给了 2 m 的横向偏差
RefTraj = zeros(3,1);            % 参考点的状态
Delta_x = zeros(3,1);            % 车辆当前状态与参考点状态的偏差
Q =[10 0 0;0 10 0;0 0 100];      % 对状态量误差的权重矩阵
R =[10];                         % 对控制量的权重矩阵
Pk =[1 0 0;0 1 0;0 0 1];         % 人为给定,相当于 Q0
Vk =[0 0 0]';
```

根据参考路径计算车辆控制的前馈控制量。这里假设无人驾驶车辆跟踪一段圆心在原点、半径为 20 m 的圆弧。计算前馈控制量的函数如下：


```
Function [RefTraj_x,RefTraj_y,RefTraj_theta,RefTraj_
delta]=
Func_CircularReferenceTrajGenerate(Pos_x,Pos_y,CEN_x,
CEN_y,Radius,N,Velo,Ts,L)
% RefTraj 为要生成的参考路径,Pos_x,Pos_y 为车辆坐标,CEN_x,
CEN_y,Radius 为圆心与半径
% N 要生成几个参考点,即预测空间,Velo,Ts 车速与采样时间,L 汽车的轴距
RefTraj = zeros(N,4);            % 生成的参考路径
Alpha_init = Func_Alpha_Pos(CEN_x,CEN_y,Pos_x,Pos_y);
                                  % 首先根据车辆位置和圆心确定 alpha
Omega = Velo/Radius              % 已知车速和半径,可以求得角速度
DFWA = atan(L/Radius);
for k = 1:1:N
    Alpha(k)= Alpha_init + Omega * Ts * (k-1);
    RefTraj(k,1)= Radius * cos(Alpha(k))+ CEN_x;% x
    RefTraj(k,2)= Radius * sin(Alpha(k))+ CEN_y;% y
    RefTraj(k,3)= Func_Theta_Pos(Alpha(k));% theta
    RefTraj(k,4)= DFWA;    % 前轮偏角,可以当作前馈量
end
```

```
RefTraj_x=RefTraj(:,1);
RefTraj_y=RefTraj(:,2);
RefTraj_theta=RefTraj(:,3);
RefTraj_delta=RefTraj(:,4);
end
```

然后计算消除跟踪误差的反馈控制量，计算公式如式（3.49）所示。算法实现如下所示，具体的 Matlab 实现 m 文件请参考 Chapter_ 3_ 5_ 3. m。

```
for i=1:1:N_1
    G_Test=3;% 跟踪第几个参考点,可以应对系统的延时
    % 先确定参考点和确定矩阵A,B.这里姑且认为A和B是不变的
[RefTraj_x,RefTraj_y,RefTraj_theta,RefTraj_delta]=Func
_CircularReferenceTrajGenerate(x_real(1,i),x_real
        (2,i),CEN(1),CEN(2),Radius,Hp,vel,T,L);
    u_feedForward=RefTraj_delta(G_Test);% 前馈控制量
    Delta_x(1,1)=x_real(1,i)-RefTraj_x(G_Test);
    Delta_x(2,1)=x_real(2,i)-RefTraj_y(G_Test);
    Delta_x(3,1)=x_real(3,i)-RefTraj_theta(G_Test);
    if  Delta_x(3,1)>pi  % 限制航向偏差的范围
        Delta_x(3,1)=Delta_x(3,1)-2*pi;
    else if Delta_x(3,1)<-1*pi
            Delta_x(3,1)=Delta_x(3,1)+2*pi;
        else
            Delta_x(3,1)=Delta_x(3,1);
        end
    end
    % 通过Backward recursion求K
    for j=Hp:-1:2
        Pk_1=Pk;
        Vk_1=Vk;
        A=[1 0 -vel*sin(RefTraj_theta(j-1))*T;0 1 vel*
        cos(RefTraj_theta(j-1))*T;0 0 1;];
        COS2=cos(RefTraj_delta(j-1))^2;
```

```
            B=[0 0  vel*T/(L*COS2)]';
            K=(B'*Pk_1*A)/(B'*Pk_1*B+R);
            Ku=R/(B'*Pk_1*B+R);
            Kv=B'/(B'*Pk_1*B+R);
            Pk=A'*Pk_1*(A-B*K)+Q;
            Vk=(A-B*K)'*Vk_1-K'*R*RefTraj_delta(j-1);
        end
        u_feedBackward=-K*(Delta_x)-Ku*u_feedForward-Kv
*Vk_1;
        FWA(i+1,1)=u_feedForward+u_feedBackward;
        [x_real(1,i+1),x_real(2,i+1),x_real(3,i+1)]=Func_
           VehicleKineticModule_Euler(x_real(1,i),x_real(2,
           i),x_real(3,i),vel,FWA(i,1),FWA(i+1,1),T,L);
    End
```


其中，$Q(t)$ 通常是对角线常阵，对角线上的元素分别表示对应相应误差分量的重视程度。越被重视的误差分量，希望它越小，相应的其加权系数就应取得越大。如果对误差在动态过程中不同阶段有不同的强调，那么相应地就应取成时变的。基于 LQR 的路径跟踪效果如图 3.18 所示。

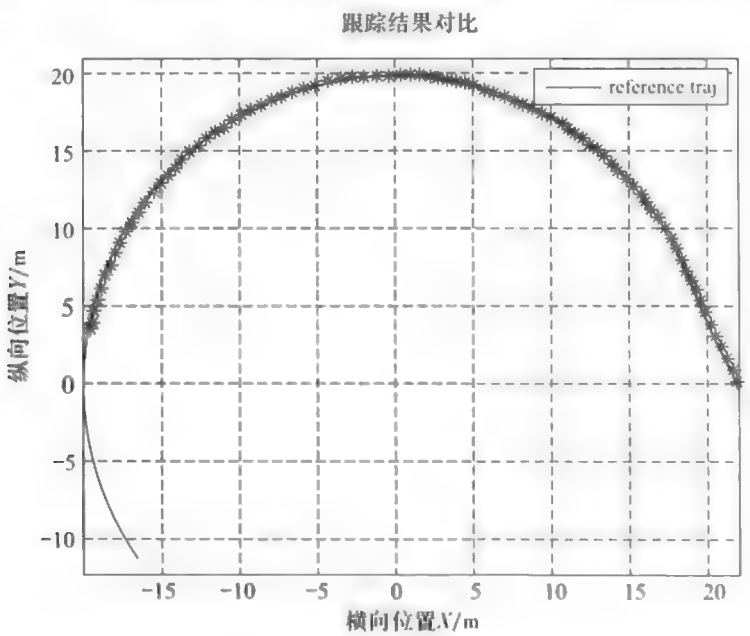


图 3.18　线性约束下基于 LQR 的路径跟踪

3.5.4　小结

对于线性系统的控制器设计问题，如果其性能指标是状态变量和（或）控制变量的二次型函数的积分，则这种动态系统的最优化问题就被称为线性系统二次型性能指标的最优控制问题，简称线性二次型最优控制问题或线性二次型问题。线性二次型问题的最优解可以写成统一的解析表达式，实现求解过程的规范化，并可简单地采用状态线性反馈控制律构成闭环最优控制系统，能够兼顾多项性能指标，因此得到特别的重视，为现代控制理论中发展较为成熟的一部分。LQR 最优控制利用廉价成本可以使原系统达到较好的性能指标（事实也可以对不稳定的系统进行整定），而且方法简单，便于实现，同时利用 Matlab 强大的功能体系容易对系统实现仿真。本节详细介绍了线性二次型规划器 LQR，并介绍了在车辆运动学模型的基础上的 LQR 路径跟踪控制器，给出了函数实现代码。

第4章

给定轨迹的轨迹跟踪控制

本章首先对无人驾驶车辆的轨迹跟踪问题进行了描述，结合模型预测控制算法的特点提出了基于模型预测控制的轨迹跟踪问题的流程以及控制结构。由于无人驾驶车辆在不同速度下对轨迹跟踪问题的要求不同，分别基于运动学模型和动力学模型设计了两类轨迹跟踪控制器。其中，基于动力学模型的跟踪控制器采用车辆横向动力学模型，对车辆进行主动转向控制。设计过程中，结合第2章中的模型以及第3章的算法推导，从线性误差模型、约束条件以及控制器目标函数选择3方面进行设计。

为了验证所设计控制系统的有效性与可靠性，选择成熟的商用动力学仿真软件CarSim与Matlab/Simulink构建联合仿真平台。参考现有无人驾驶车辆平台的整车配置与参数，在CarSim中搭建了整车模型，将控制算法封装在自定义的S函数中，实现了无人驾驶车辆在不同环境中的轨迹跟踪仿真。

最后，分别在直线参考轨迹和圆形参考轨迹下对基于运动学模型的轨迹跟踪控制器进行仿真。仿真结果表明，所设计的控制器能够快速且稳定地跟踪期望轨迹。

4.1 问题的描述

无人驾驶车辆的轨迹跟踪问题是指根据某种控制理论，为系统设计一个控制输入作用，使无人驾驶车辆能够到达并最终以期望的速度跟踪期望轨

迹。在惯性坐标系中，车辆必须从一个给定的初始状态出发。这个初始点既可以在期望轨迹上，也可以不在期望轨迹上。在任意时刻 k，无人驾驶车辆的轨迹跟踪问题可以用图 4.1 表示，其中期望轨迹用一个个离散的轨迹点给出。期望轨迹是指一条几何曲线 $f(\boldsymbol{x}_r(t))$，自变量 $\boldsymbol{x}_r$ 是时间 t 的函数，曲线方程是 t 的隐函数。给定的速度是指广义的速度变量，也即参考控制输入，包括速度、角速度和前轮偏角等，用 $\boldsymbol{u}_r$ 表示。$f(\boldsymbol{x}_r(t))$ 和 $\boldsymbol{u}_r$ 既可以由轨迹规划模块提供，也可以预先设定。在本章中，设定期望轨迹和参考控制输入已经被给出。

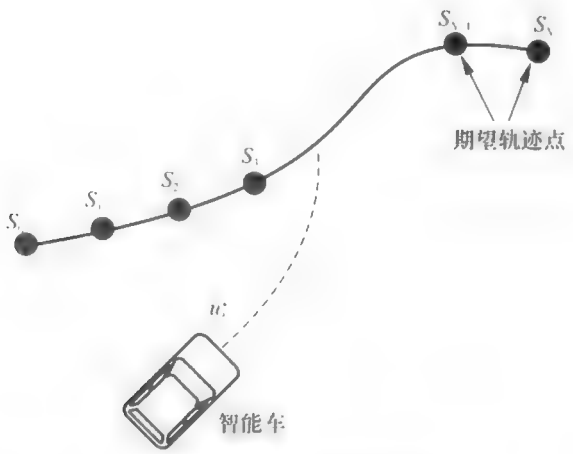


图 4.1　无人驾驶车辆轨迹跟踪示意

根据本书的研究内容，进一步对无人驾驶车辆的轨迹跟踪问题做出如下限定：车辆为前轮转向车辆，而且系统能够有效地提供两类信息：

1　可行驶区域的几何描述、路面特征及路面摩擦系数

2　车辆位置及内部状态，包括纵横向速度、加速度、轮速等参数

这就是说轨迹跟踪控制是在周围环境及车辆内部状态完全已知的情况下进行的，不涉及环境感知和车辆状态的估计。根据上述设定，基于模型预测控制的轨迹跟踪过程如图 4.2 所示。

轨迹跟踪过程中所采用的模型预测控制算法如图 4.3 所示。其中虚线框表示的是模型预测控制器的主体，主要由线性误差模型、系统约束以及目标函数组成。线性误差方程是轨迹跟踪控制系统的数学描述，也是构建控制算法的基础。系统约束包括车辆执行机构约束、控制量平滑约束以及车辆稳定性约束等。目标函数的设计则综合考虑轨迹跟踪的快速性以及平稳性。以下从这 3 方面分别介绍基于运动学模型的轨迹跟踪控制器的设计。

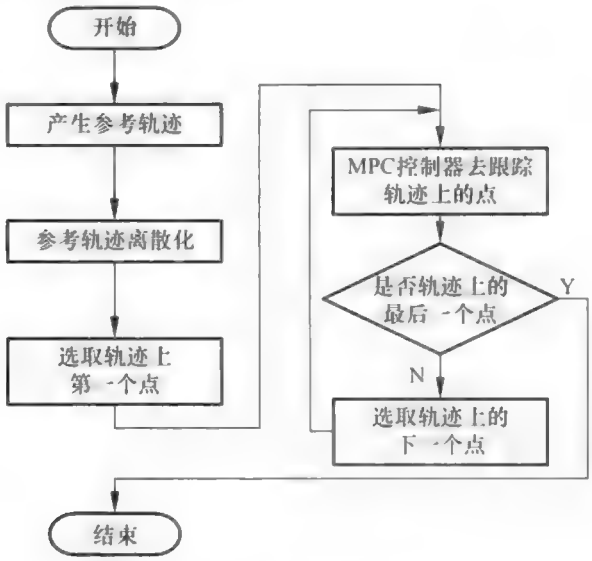


图 4.2　轨迹跟踪流程

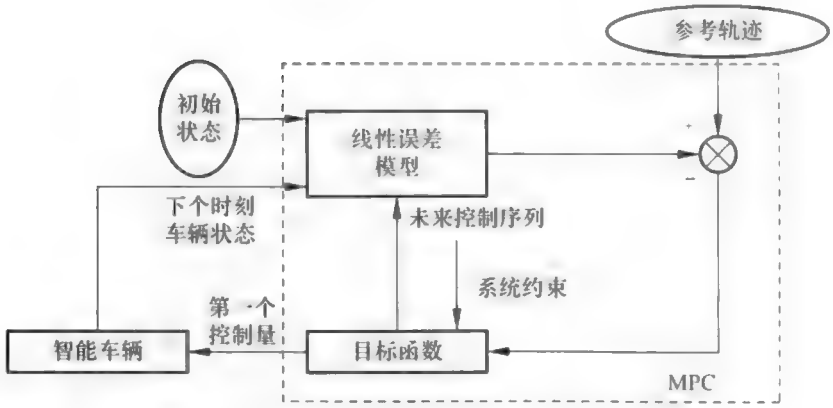


图 4.3　基于模型预测控制的轨迹跟踪控制器

4.2　基于运动学模型的轨迹跟踪控制器设计

4.2.1　车辆运动学建模

为方便阅读，将第 2 章车辆简化后的运动学模型图 2.1 移至第 4 章，如图 4.4所示。

在地面固定坐标系 OXY 下，车辆运动学方程为：

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\varphi}\end{bmatrix}=\begin{bmatrix}\cos\varphi\\ \sin\varphi\\ \dfrac{\tan\delta}{l}\end{bmatrix}v \tag{4.1}$$

式中：$(x,\ y)$ 为车辆后轴中心的坐标，φ 为车体的航向角，δ 为前轮偏角，v 为车辆后轴速度，l 为轴距。

由式（4.1）可知，系统可以被看作一个输入为 $\boldsymbol{u}(v,\ \delta)$ 和状态量为 $\boldsymbol{X}(x,\ y,\ \varphi)$ 的控制系统。其一般形式为：

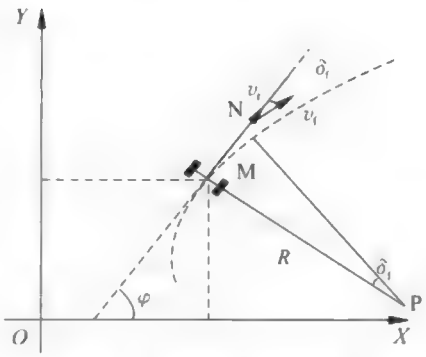


图4.4　车辆运动学模型

$$\dot{\boldsymbol{X}}=f(\boldsymbol{X},\boldsymbol{u}) \tag{4.2}$$

对于给定的参考轨迹，可以由参考车辆的运动轨迹描述。其上的每一个点都满足上述运动学方程，用r代表参考量，一般形式为：

$$\dot{\boldsymbol{X}}_r=f(\boldsymbol{X}_r,\boldsymbol{u}_r) \tag{4.3}$$

式中：$\boldsymbol{X}_r=[x_r\quad y_r\quad \varphi_r]^{\mathrm{T}}$，$\boldsymbol{u}_r=[v_r\quad \delta_r]$

将式（4.2）在参考轨迹点采用泰勒级数展开并忽略高阶项，得到：

$$\dot{\boldsymbol{X}}=f(\boldsymbol{X}_r,\boldsymbol{u}_r)+\left.\frac{\partial f(\boldsymbol{X},\boldsymbol{u})}{\partial\boldsymbol{X}}\right|_{\substack{\boldsymbol{X}=\boldsymbol{X}_r\\ \boldsymbol{u}=\boldsymbol{u}_r}}(\boldsymbol{X}-\boldsymbol{X}_r)+\left.\frac{\partial f(\boldsymbol{X},\boldsymbol{u})}{\partial\boldsymbol{u}}\right|_{\substack{\boldsymbol{X}=\boldsymbol{X}_r\\ \boldsymbol{u}=\boldsymbol{u}_r}}(\boldsymbol{u}-\boldsymbol{u}_r) \tag{4.4}$$

将式（4.4）与式（4.3）相减，可以得到：

$$\dot{\tilde{\boldsymbol{X}}}=\begin{bmatrix}\dot{x}-\dot{x}_r\\ \dot{y}-\dot{y}_r\\ \dot{\varphi}-\dot{\varphi}_r\end{bmatrix}=\begin{bmatrix}0 & 0 & -v_r\sin\varphi_r\\ 0 & 0 & v_r\cos\varphi_r\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}x-x_r\\ y-y_r\\ \varphi-\varphi_r\end{bmatrix}+\begin{bmatrix}\cos\varphi_r & 0\\ \sin\varphi_r & 0\\ \dfrac{\tan\delta_r}{l} & \dfrac{v_r}{l\cos^2\delta_r}\end{bmatrix}\begin{bmatrix}v-v_r\\ \delta-\delta_r\end{bmatrix} \tag{4.5}$$

式（4.5）即线性化的无人驾驶车辆误差模型。为了能够将该模型应用于模型预测控制器的设计，对式（4.5）进行离散化处理：

$$\tilde{\boldsymbol{X}}(k+1)=\boldsymbol{A}_{k,t}\tilde{\boldsymbol{X}}(k)+\boldsymbol{B}_{k,t}\tilde{\boldsymbol{u}}(k) \tag{4.6}$$

式中：$\boldsymbol{A}_{k,t}=\begin{bmatrix}1 & 0 & -v_r\sin\varphi_r T\\ 0 & 1 & v_r\cos\varphi_r T\\ 0 & 0 & 1\end{bmatrix}$，$\boldsymbol{B}_{k,t}=\begin{bmatrix}\cos\varphi_r T & 0\\ \sin\varphi_r T & 0\\ \dfrac{\tan\delta_r T}{l} & \dfrac{v_r T}{l\cos^2\delta_r}\end{bmatrix}$，$T$ 为采样时间。

4.2.2 目标函数设计

目标函数要能够保证无人驾驶车辆快速且平稳地追踪期望轨迹，就需要加入对系统状态量的偏差和控制量的优化。文献［50］在设计轨迹跟踪控制器时，采用如下形式的目标函数：

$$J(k) = \sum_{j=1}^{N} \tilde{\boldsymbol{X}}^{\mathrm{T}}(k+j \mid k)\boldsymbol{Q}\tilde{\boldsymbol{X}}(k+j) + \tilde{\boldsymbol{u}}^{\mathrm{T}}(k+j-1)\boldsymbol{R}\tilde{\boldsymbol{u}}(k+j-1) \tag{4.7}$$

式中，$\boldsymbol{Q}$ 和 $\boldsymbol{R}$ 为权重矩阵。

第一项反映了系统对参考轨线的跟随能力，第二项反映了对控制量变化的约束。该目标函数的优点在于容易转换成标准二次规划形式，但是也存在比较明显的缺陷：无法对每个采样周期内的控制增量进行限制，即无法避免被控系统控制量突变的现象，从而影响控制量的连续性。参照文献［72］所使用的软约束方法，采用如下形式的目标函数：

$$J(k) = \sum_{i=1}^{N_p} \| \boldsymbol{\eta}(k+i \mid t) - \boldsymbol{\eta}_{ref}(k+i \mid t) \|_{\boldsymbol{Q}}^2 + \sum_{i=1}^{N_c-1} \| \Delta \boldsymbol{U}(k+i \mid t) \|_{\boldsymbol{R}}^2 + \rho\varepsilon^2 \tag{4.8}$$

式中，N_p 为预测时域；N_c 为控制时域；ρ 为权重系数；ε 为松弛因子。

相比式（4.7），式（4.8）用控制增量取代控制量并且加入了松弛因子，这样不仅能对控制增量进行直接的限制，也能防止执行过程中出现没有可行解的情况。在目标函数中，需要计算未来一段时间系统的输出，以下重点介绍如何基于无人驾驶车辆线性误差模型预测车辆未来时刻的输出：

将式（4.6）做如下转换：

$$\boldsymbol{\xi}(k \mid t) = \begin{bmatrix} \tilde{\boldsymbol{X}}(k \mid t) \\ \tilde{\boldsymbol{u}}(k-1 \mid t) \end{bmatrix} \tag{4.9}$$

得到一个新的状态空间表达式：

$$\begin{aligned} \boldsymbol{\xi}(k+1 \mid t) &= \tilde{\boldsymbol{A}}_{k,t}\boldsymbol{\xi}(k \mid t) + \tilde{\boldsymbol{B}}_{k,t}\Delta \boldsymbol{U}(k \mid t) \\ \boldsymbol{\eta}(k \mid t) &= \tilde{\boldsymbol{C}}_{k,t}\boldsymbol{\xi}(k \mid t) \end{aligned} \tag{4.10}$$

式中，$\tilde{\boldsymbol{A}}_{k,t} = \begin{bmatrix} \boldsymbol{A}_{k,t} & \boldsymbol{B}_{k,t} \\ \boldsymbol{0}_{m \times n} & \boldsymbol{I}_m \end{bmatrix}$，$\tilde{\boldsymbol{B}}_{k,t} = \begin{bmatrix} \boldsymbol{B}_{k,t} \\ \boldsymbol{I}_m \end{bmatrix}$，$n$ 为状态量维度，m 为控制量维度。

为了简化计算，本书做出如下假设：

$$\boldsymbol{A}_{k,t} = \boldsymbol{A}_{t,t}, \quad k = 1, \cdots, t+N-1$$

$$\boldsymbol{B}_{k,t} = \boldsymbol{B}_{t,t}, \quad k = 1, \cdots, t + N - 1 \tag{4.11}$$

经过推导，可以得到系统的预测输出表达式：

$$\boldsymbol{Y}(t) = \boldsymbol{\Psi}_t \boldsymbol{\xi}(t \mid t) + \boldsymbol{\Theta}_t \Delta \boldsymbol{U}(t) \tag{4.12}$$

式中：

$$\boldsymbol{Y}(t) = \begin{bmatrix} \boldsymbol{\eta}(t+1 \mid t) \\ \boldsymbol{\eta}(t+2 \mid t) \\ \cdots \\ \boldsymbol{\eta}(t+N_c \mid t) \\ \cdots \\ \boldsymbol{\eta}(t+N_p \mid t) \end{bmatrix} \quad \boldsymbol{\Psi}_t = \begin{bmatrix} \tilde{\boldsymbol{C}}_{t,t}\tilde{\boldsymbol{A}}_{t,t} \\ \tilde{\boldsymbol{C}}_{t,t}\tilde{\boldsymbol{A}}_{t,t}^2 \\ \cdots \\ \tilde{\boldsymbol{C}}_{t,t}\tilde{\boldsymbol{A}}_{t,t}^{N_c} \\ \cdots \\ \tilde{\boldsymbol{C}}_{t,t}\tilde{\boldsymbol{A}}_{t,t}^{N_p} \end{bmatrix}$$

$$\boldsymbol{\Theta}_t = \begin{bmatrix} \boldsymbol{C}_{t,t}\tilde{\boldsymbol{B}}_{t,t} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \tilde{\boldsymbol{C}}_{t,t}\tilde{\boldsymbol{A}}_{t,t}\tilde{\boldsymbol{B}}_{t,t} & \tilde{\boldsymbol{C}}_{t,t}\tilde{\boldsymbol{B}}_{t,t} & \boldsymbol{0} & \boldsymbol{0} \\ \cdots & \cdots & \ddots & \cdots \\ \tilde{\boldsymbol{C}}_{t,t}\tilde{\boldsymbol{A}}_{t,t}^{N_c-1}\tilde{\boldsymbol{B}}_{t,t} & \tilde{\boldsymbol{C}}_{t,t}\tilde{\boldsymbol{A}}_{t,t}^{N_c-2}\tilde{\boldsymbol{B}}_{t,t} & \cdots & \tilde{\boldsymbol{C}}_{t,t}\tilde{\boldsymbol{B}}_{t,t} \\ \tilde{\boldsymbol{C}}_{t,t}\tilde{\boldsymbol{A}}_{t,t}\tilde{\boldsymbol{B}}_{t,t} & \tilde{\boldsymbol{C}}_{t,t}\tilde{\boldsymbol{A}}_{t,t}\tilde{\boldsymbol{B}}_{t,t} & \cdots & \tilde{\boldsymbol{C}}_{t,t}\tilde{\boldsymbol{A}}_{t,t}\tilde{\boldsymbol{B}}_{t,t} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\boldsymbol{C}}_{t,t}\tilde{\boldsymbol{A}}_{t,t}^{N_p-1}\tilde{\boldsymbol{B}}_{t,t} & \tilde{\boldsymbol{C}}_{t,t}\tilde{\boldsymbol{A}}_{t,t}^{N_p-2}\tilde{\boldsymbol{B}}_{t,t} & \cdots & \tilde{\boldsymbol{C}}_{t,t}\tilde{\boldsymbol{A}}_{t,t}^{N_p-N_c-1}\tilde{\boldsymbol{B}}_{t,t} \end{bmatrix}, \Delta \boldsymbol{U}(t) = \begin{bmatrix} \Delta \boldsymbol{u}(t \mid t) \\ \Delta \boldsymbol{u}(t+1 \mid t) \\ \cdots \\ \Delta \boldsymbol{u}(t+N_c \mid t) \end{bmatrix}$$

将式（4.12）代入式（4.8），即可得到完整形式的目标函数

4.2.3　约束条件设计

本书主要考虑控制过程中的控制量极限约束和控制增量约束。控制量表达形式为：

$$\boldsymbol{u}_{\min}(t+k) \leqslant \boldsymbol{u}(t+k) \leqslant \boldsymbol{u}_{\max}(t+k), \quad k = 0, 1, \cdots, N_c - 1 \tag{4.13}$$

控制增量约束表达形式为：

$$\Delta \boldsymbol{u}_{\min}(t+k) \leqslant \Delta \boldsymbol{u}(t+k) \leqslant \Delta \boldsymbol{u}_{\max}(t+k), \quad k = 0, 1, \cdots, N_c - 1 \tag{4.14}$$

在目标函数中，求解的变量为控制时域内的控制增量，约束条件中也只能以控制增量或者是控制增量与转换矩阵相乘的形式出现。因此，需要对式（4.14）进行转换，求得相应的转换矩阵。因为存在如下关系：

$$u(t+k) = u(t+k-1) + \Delta u(t+k) \tag{4.15}$$

设：

$$U_t = \mathbf{1}_{N_c} \otimes u(k-1) \tag{4.16}$$

$$A = \underbrace{\begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix}}_{N_c \times N_c} \otimes I_m \tag{4.17}$$

式中：$\mathbf{1}_{N_c}$是行数为 N_c 的列向量，I_m 是维度为 m 的单位矩阵，$\otimes$为克罗内克积（Kronecker product），$u(k-1)$ 为上一时刻实际的控制量。

结合式（4.15）~式(4.17)，可以将式（4.13）转换为以下形式：

$$U_{\min} \leqslant A\Delta U_t + U_t \leqslant U_{\max} \tag{4.18}$$

式中：$U_{\min}$，$U_{\max}$分别为控制时域内的控制量最小值、最大值集合。

将目标函数转化为标准二次型形式并结合约束条件，解决以下优化问题：

$$J(\xi(t), u(t-1), \Delta U(t)) = [\Delta U(t)^{\mathrm{T}}, \varepsilon]^{\mathrm{T}} H_t [\Delta U(t)^{\mathrm{T}}, \varepsilon] + G_t [\Delta U(t)^{\mathrm{T}}, \varepsilon]^{\mathrm{T}} \tag{4.19}$$

$$s.t.\ \Delta U_{\min} \leqslant \Delta U_t \leqslant \Delta U_{\max}$$

$$U_{\min} \leqslant A\Delta U_t + U_t \leqslant U_{\max}$$

式中：$H_t = \begin{bmatrix} \Theta_t^{\mathrm{T}} Q \Theta_t + R & 0 \\ 0 & \rho \end{bmatrix}$，$G_t = \begin{bmatrix} 2e_t^{\mathrm{T}} Q \Theta_t & 0 \end{bmatrix}$，$e_t$ 为预测时域内的跟踪误差。

在每一控制周期内完成对式（4.19）的求解后，得到了控制时域内的一系列控制输入增量：

$$\Delta U_t^* = [\Delta u_t^*, \Delta u_{t+1}^*, \cdots, \Delta u_{t+N_c-1}^*]^{\mathrm{T}} \tag{4.20}$$

将该控制序列中第一个元素作为实际的控制输入增量作用于系统，即：

$$u(t) = u(t-1) + \Delta u_t^* \tag{4.21}$$

进入下一个控制周期后，重复上述过程，如此循环实现了对车辆的轨迹跟踪控制。

约束条件中涉及的具体数值则需要通过下面介绍的纵横向跟踪能力测试获得。

在无人驾驶车辆纵向跟踪能力实验中，分别对无人驾驶车辆输入 5 m/s、7.5 m/s 和 10 m/s 的阶跃期望速度，平稳行驶一段时间后再以最大制动能力对车辆进行制动，记录加速和减速时间，测试结果如表 4.1 所示。可以看出，车辆在中低速（7.5 m/s 以下）加减速过程较为平稳，加速度维持在 1 m/s² 左

行。为了使跟踪过程更加平稳，无人驾驶车辆速度控制约束可设置为：

$$\begin{aligned} -0.2\ \text{m/s} \leqslant v - v_d \leqslant 0.2\ \text{m/s} \\ -0.05\ \text{m/s} \leqslant \Delta v \leqslant 0.05\ \text{m/s} \end{aligned} \tag{4.22}$$

其中，v_d 为车辆期望速度，Δv 为每个控制周期的速度增量。

表 4.1　无人驾驶车辆纵向跟踪能力测试

项目	加速时间 t_1/s	制动时间 t_2/s
0～5 m/s	5	6
0～7.5 m/s	8	8
0～10 m/s	15	12

在横向跟踪能力实验中，将车辆前轮从左极限位置连续转向至右极限位置，记录相应的位置和时间，测试结果如表 4.2 所示。通过计算可知，方向盘转一圈大约需要 1.8 s，对应于前轮偏角为 17°。因此，将车辆极限前轮偏角和前轮偏角增量设置为：

$$\begin{aligned} -25° \leqslant \delta \leqslant 25° \\ -0.47° \leqslant \Delta\delta \leqslant 0.47° \end{aligned} \tag{4.23}$$

表 4.2　无人驾驶车辆原地转向性能测试

项目	逆时针方向	顺时针方向
方向盘极限转角 α/(°)	-511	492
对应前轮偏角 δ_f/(°)	25	25
所用时间 t/s	4.9	4.9

结合式（4.22）和式（4.23），可以得到控制量的约束条件为：

控制量约束：$$\begin{bmatrix} -0.2 \\ -25 \end{bmatrix} \leqslant \boldsymbol{u}_{\text{kin}} \leqslant \begin{bmatrix} 0.2 \\ 25 \end{bmatrix}$$

控制增量约束：$$\begin{bmatrix} -0.05 \\ -0.47 \end{bmatrix} \leqslant \Delta\boldsymbol{u}_{\text{kin}} \leqslant \begin{bmatrix} 0.05 \\ 0.47 \end{bmatrix}$$

4.3　仿真平台概述

本章所构建的联合仿真平台是包含车辆动力学模拟、控制系统建模和动画演示的统一平台。目前，控制系统的建模与开发大多是在 Matlab/Simulink 环境中完成的。在该环境中，开发人员可以利用图形建模功能编写复杂的控制逻辑；同时，世界上主流动力学仿真与分析软件都提供了与 Simulink 仿真的接口。所以，我们选择 Simulink 环境进行控制系统建模。

为了满足车辆动力学模型精度的要求，快速有效地验证轨迹跟踪控制算法，需要选用成熟并且具备高精度的车辆动力学仿真软件。符合条件的整车动力学仿真软件主要包括美国 MSC 公司的 CarSim，德国 dSPACE 公司的 ASM，加拿大 CM－LABS 公司的 Vortex 等。由于 CarSim 同时具备了车辆动力学仿真与三维动画演示功能，其扩展包 CarSim RT 提供了与硬件实时系统的接口，十分适合无人驾驶车辆控制系统的开发与测试。因此，选用 CarSim 作为车辆动力学仿真软件，版本为 8.02。

综合上述分析，联合仿真平台是以汽车动力学仿真软件 CarSim 和控制系统仿真软件 Matlab/Simulink 为基础开发的。为了充分结合两者的优势，需要在 CarSim 中建立准确的整车高精度模型，在 Simulink 中搭建控制器。最后，通过系统的输出和性能指标验证运动规划和控制算法的各项性能。

4.3.1 CarSim 软件介绍

CarSim 是由美国 MSC 公司（Mechanical Simulation Corporation，MSC）开发的车辆动力学仿真软件。它的动力学模型基础来源于密歇根大学交通运输研究所（University of Michigan Transportation Research Institute，UMTRI）多年理论和实践经验的积累。CarSim 可以方便灵活地定义试验环境和试验过程，准确预测和仿真汽车整车的操纵稳定性、动力性、平顺性等，适用于轿车、轻型货车等车型的建模仿真。其主要功能界面如图 4.5 所示。从图 4.5 可以看出，CarSim 主要包含 3 大功能模块：①模型和工况参数设定；②数学模型求解；③输出和后处理。

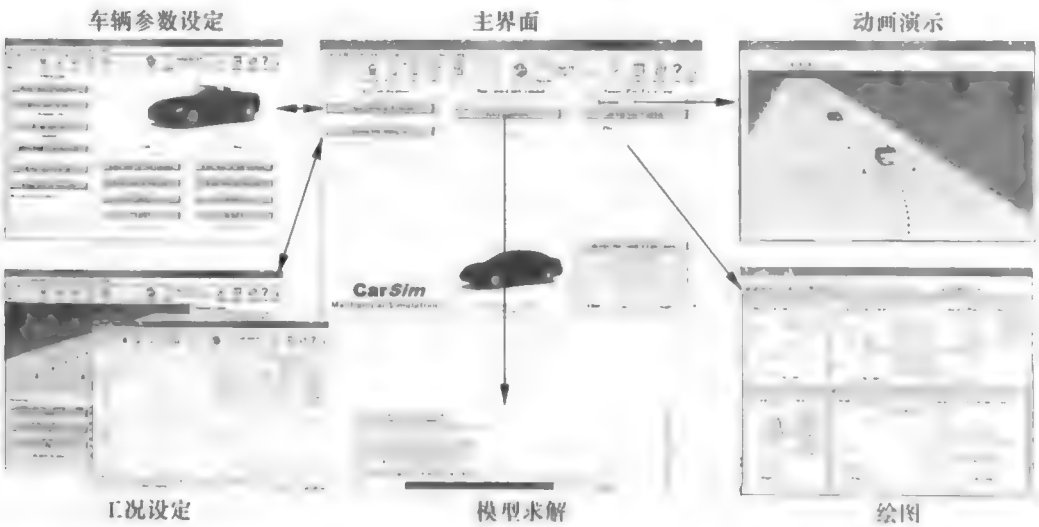


图 4.5 CarSim 功能与界面示意

（1）模型和工况参数设定

该功能模块主要包括车辆参数设定与车辆测试工况设定两部分，其中车辆参数设定需要从已有的整车模型数据库中选择合适的车型，并对动力传动系统、悬架系统等进行设置。车辆测试工况设定主要是对车辆行驶环境及相关因素进行设置，包括路面信息、风阻等外部环境信息。

（2）数学模型求解

该功能模块是整个软件求解运算的内核，需要设置求解器类型、仿真步长和仿真时间等信息；同时，该模块也是与其他所有外部环境的接口，包括Simulink，Labview 和 dSPACE 等。

（3）输出和后处理部分

该功能模块包括仿真结果的3D动画显示和数据绘图两部分。用户可以通过仿真动画窗口直观地观察汽车动力学响应，还可以在数据绘图部分有选择地输出指定参数的曲线，进行定量分析。

4.3.2　Simulink/CarSim 联合仿真平台

（1）整车模型搭建

搭建仿真平台的第一步是在 CarSim 中搭建整车模型。CarSim 采用多体动力学建模方法，对车辆进行了适当的抽象简化。在模型搭建的时候只需要依据目标车型参数依次配置车辆各子系统参数，如传动系统、制动系统等。其详细构成如图4.6所示。

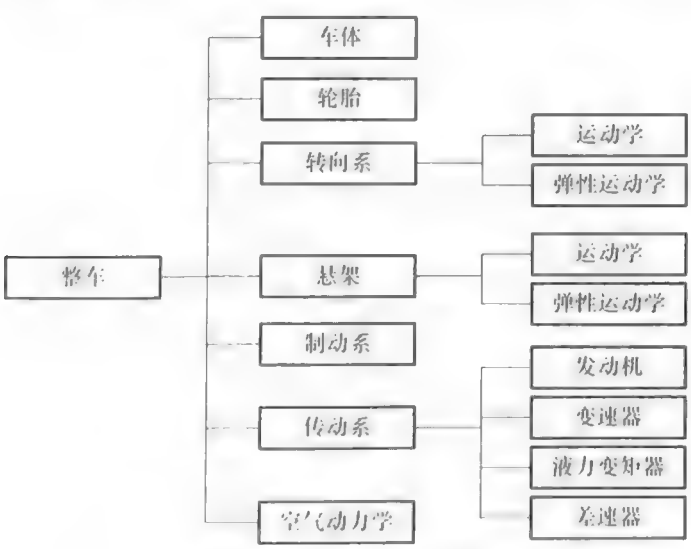


图4.6　CarSim 整车模型包括的子系统

为了方便后续控制算法向实车实验的移植，尽量贴近实际情况，以北京理工大学智能车辆研究所现有的无人驾驶车辆平台——比亚迪速锐 2012 款 1.5TID 自动旗舰型作为目标车型，对上述 7 个子系统进行设置，具体配置如表 4.3 所示。

表 4.3　车辆模型主要配置

系统名称	配置类型	参　数
车体	车体结构	三厢
转向系	助力类型	电动助力
悬架	前悬架	独立悬架
	后悬架	扭力梁
制动系	ABS 配置	有
传动系	发动机	150 kW，直列 4 缸
	变速箱	6 速自动
轮胎	前轮胎规格	205/55 R16
	后轮胎规格	205/55 R16

完成系统配置后，还需要对每项配置输入相应的参数，以实现对系统的准确描述。涉及的动力学模型参数如表 4.4 所示。

表 4.4　动力学模型基本参数

动力学模型参数	符　号	数　值	单　位
整车质量	m	1 395	kg
绕 Z 轴转动惯量	I_z	4 192	$kg \cdot m^2$
质心至前轴距离	a	1.04	m
质心至后轴距离	b	1.62	m
质心高度	H	0.54	m
轮距	d	1.52	m
滚动阻力系数	f	0.02	—
轮胎滚动半径	r	0.335	m
迎风面积	A	1.8	m^2
空气阻力系数	C_D	0.343	—
空气密度	ρ	1.206	$kg \cdot m^3$

（2）仿真工况设定

仿真工况设定主要包括初始速度、驾驶员制动模式、转向模式、挡位输入

和路面参数等。由于所设计的轨迹跟踪控制器实时向车辆提供车轮转角信息和速度信息，并且加入了速度控制模块，因此制动模式和转向模式均选择为无，即不需要模型自身施加制动力和转向力，挡位控制选择自动升挡。路面参数需要道路曲线、宽度、摩擦系数等信息。

（3）外部接口设定

主要包括连接器选择、解算器选择和输入输出接口选择3部分内容。需要在Simulink环境中进行仿真，因此首先选择基于Simulink的连接器，然后设定解算器。求解算法选择龙格－库塔数值解法，设定模型仿真步长为0.001 s。输入端口包括车辆纵向速度和车轮转角，输出端口包括车辆位置信息、纵横向速度等运动学信息，以及横摆角、质心侧偏角等动力学信息。完成上述设定后，通过外部接口将车辆模型发送至指定路径下的Simulink仿真文件中。CarSim模块即以S函数的形式增加到Simulink模型库中。通过调用该S函数，并加入控制器模块和外部输入信息，即完成联合仿真环境的搭建，如图4.7所示。

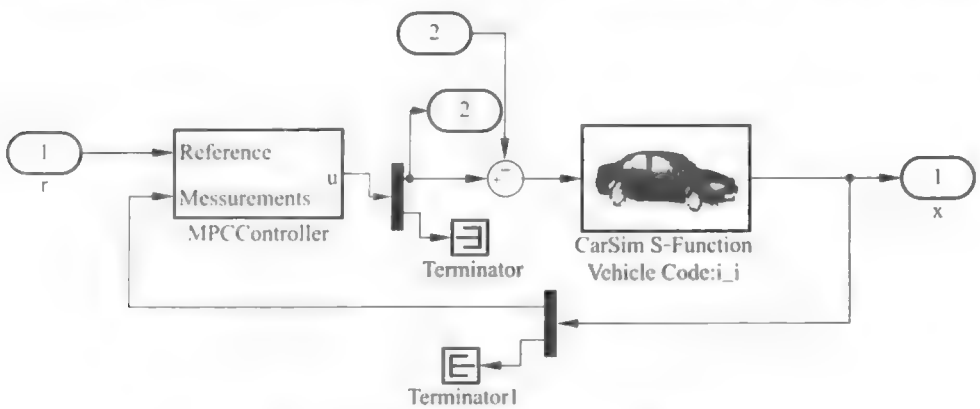


图4.7　Simulink/CarSim联合仿真平台

4.4　仿真实例

本节实例主要讲解如何通过搭建Simulink/CarSim联合仿真平台，对所设计的控制器进行仿真验证，通过实践加深对理论的理解。图4.8是对基于MPC的给定轨迹的轨迹跟踪控制器进行CarSim与Simulink联合仿真的实例。

选用CarSim作为车辆动力学仿真软件，版本为8.02。需要注意的是，选用的控制系统仿真软件Matlab/Simulink版本不能过高，否则CarSim模块无法通过外部接口增加到Simulink模块库中。高版本的Matlab如何与CarSim联合仿真，

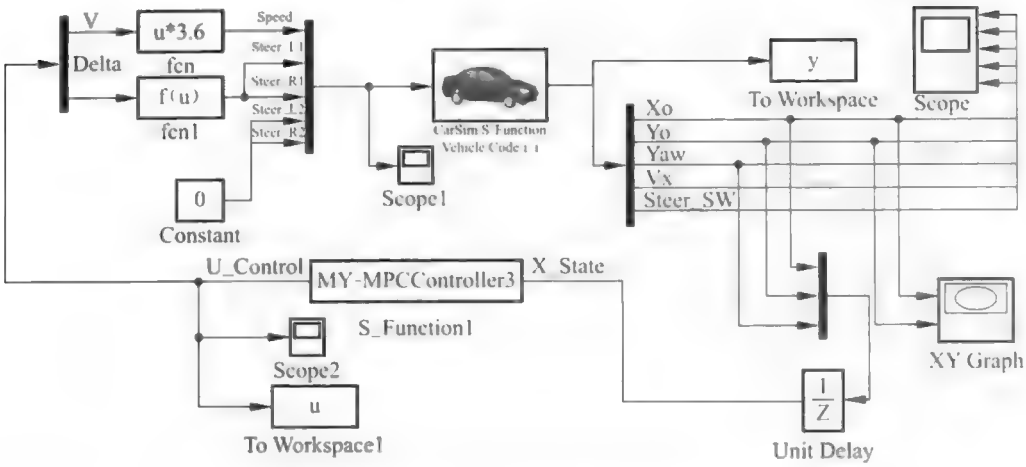


图 4.8　Simulink/CarSim 联合仿真平台

有特殊的设置方法，详见附 A　本文选用的 Matlab/Simulink 版本为 2011a

4.4.1　CarSim 与 Simulink 联合仿真

① 运行 CarSim。这里选用的 CarSim 的版本为 8.02

② 出现“Select Recent Database”（“选择数据库”）对话框，如图 4.9 所示；选择“D:\Users\Public\Documents\CarSim_Data”数据库文件（注意数据库文件的位置与自己安装过程有关），点击“Continue with the selected database”。

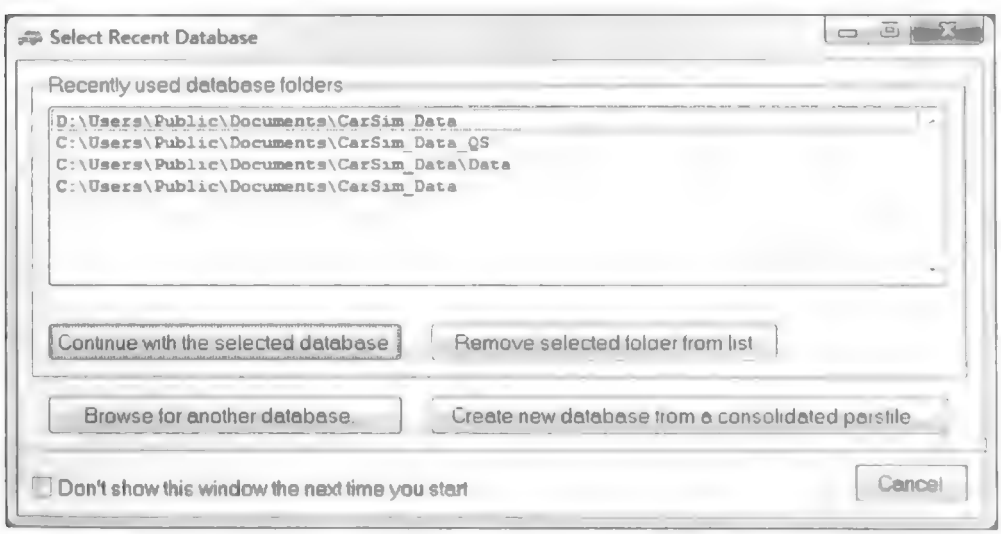


图 4.9　数据库选择

3 出现“License Settings”（“许可设置”）对话框，如图4.10所示；选择“Select”，即可打开CarSim的主界面，如图4.11所示。

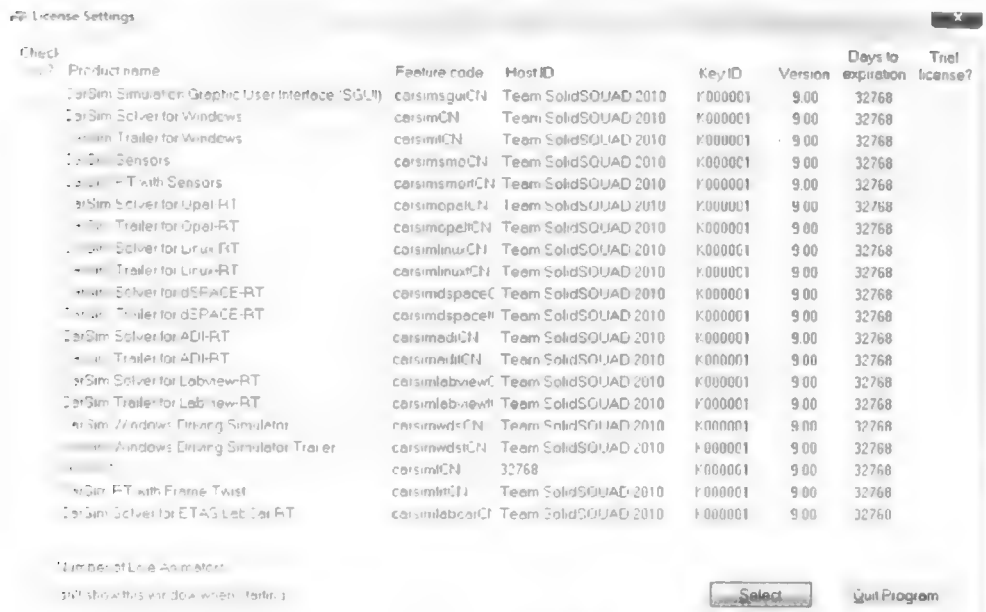


图4.10　许可设置

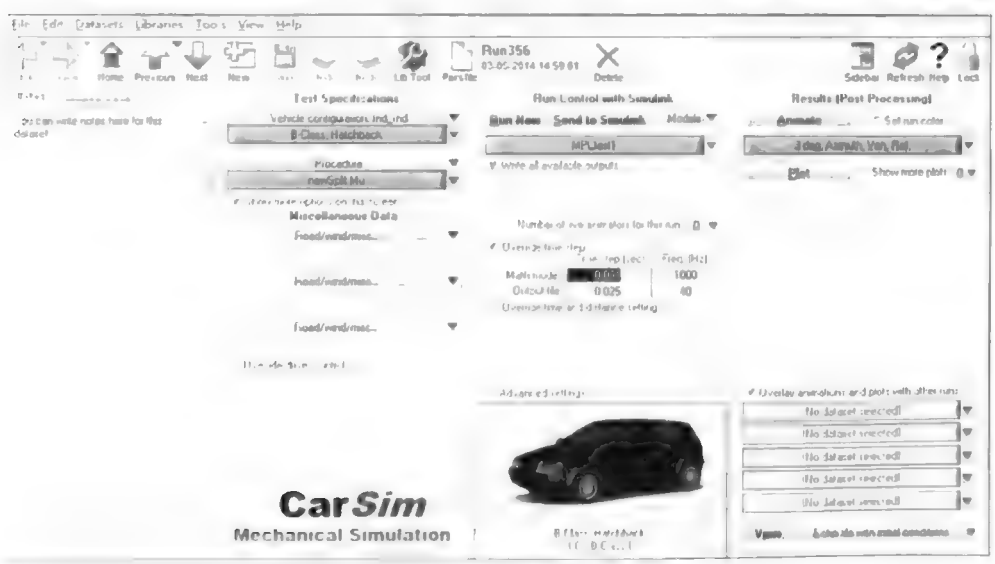


图4.11　CarSim主界面

CarSim主界面主要由3大部分组成：车辆参数及仿真工况的设置、数学模型求解与后处理。用户可以修改车辆参数，根据需要来设置仿真工况。

④ 点击“New”，新建一组 Dataset，如图 4.12 所示；在两个文本框中分别输入“Example”和“MPCtest1”，然后点击“set”，完成新建

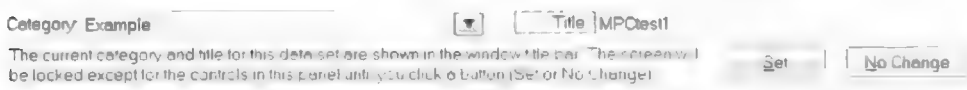


图 4.12　新建 Dataset

⑤ 选择主菜单中的“Datasets”下拉菜单（如图 4.13 所示），会发现 Example 一栏中多出“MPCTest1”。

图 4.13　第 4 步中新建的 Dataset

⑥ 点击图 4.14（a）中的下三角，选择相应的车型　选择 CS B – Class 中的 Hatchback 车型，如图 4.14 所示。

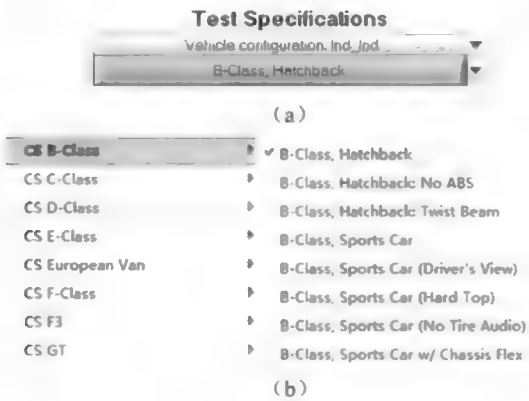


图 4.14　选择车型

⑦ 新建满足要求的仿真工况。

ⓐ 点击如图 4.15 所示新建工况，即出现如图 4.16 所示的界面

ⓑ 新建仿真工况，点击“New”，在图 4.17中的文本框中依次输入“MPC Example”和“newSplit Mu”，点击“Set”，完成新建。

图 4-15　仿真工况

ⓒ 根据要求设置仿真工况：

➢ 目标车速为 18 km/h；

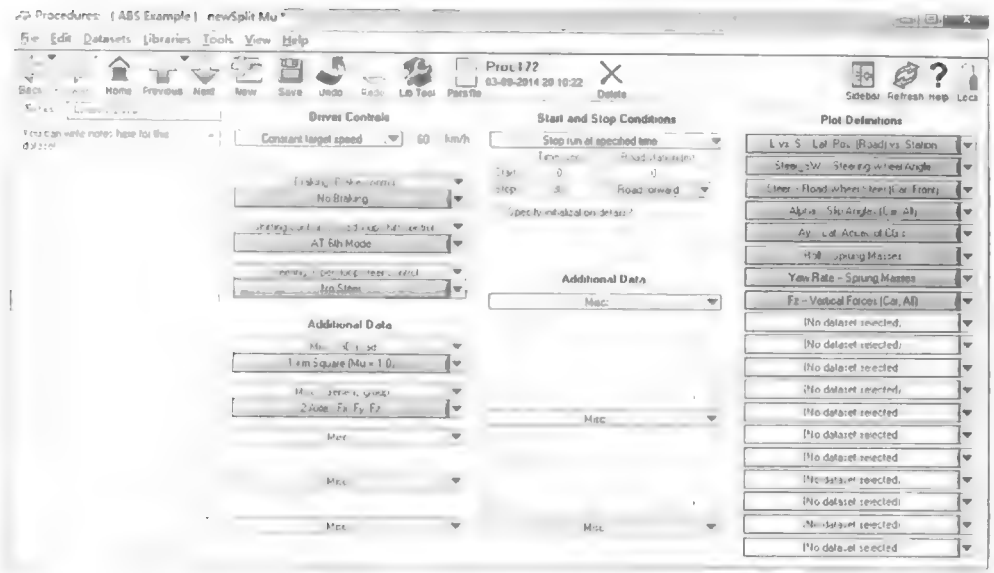


图 4.16　仿真工况的主界面

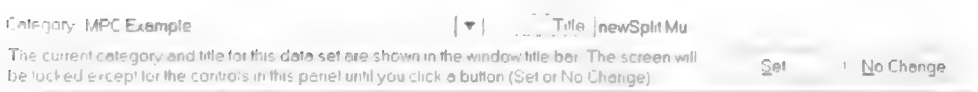


图 4.17　新建仿真工况

- 无制动；
- 挡位控制选用闭环 AT 6 挡模式；
- 无转向；
- 路面选择为 1 km^2 摩擦系数为 1.0 的方形路面。
- 完成后的设置，如图 4.18 所示。

ⓓ 设置仿真时间。

同样在设置仿真工况的主界面，在如图 4.19所示的文本框内输入“30”。

ⓔ 选择前面新建的仿真工况。

点击“Home”图标，返回 CarSim 的主界面，选择前面新建的“Split Mu”工况。

ⓕ 设置仿真步长。

在 CarSim 主界面中，点击下拉菜单“Tools”，选择“Preference”，出现如

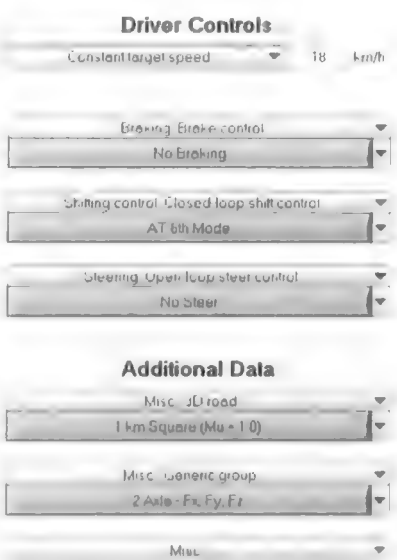


图 4.18　仿真工况设置

图 4.20所示的界面，将仿真步长设置为“0.001”。

⑧ 建立 CarSim 与 Simulink 联合仿真的模型。

ⓐ 点击如图 4.21 所示的 Models 选项，选择“Models: Simulink”

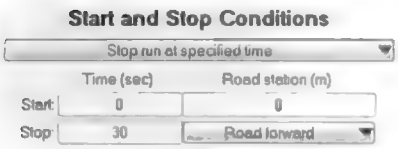


图 4.19 仿真时间设置

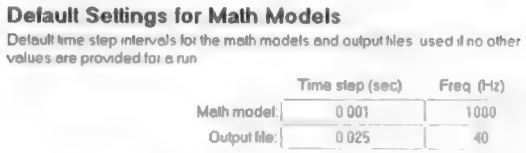


图 4.20 设置仿真步长

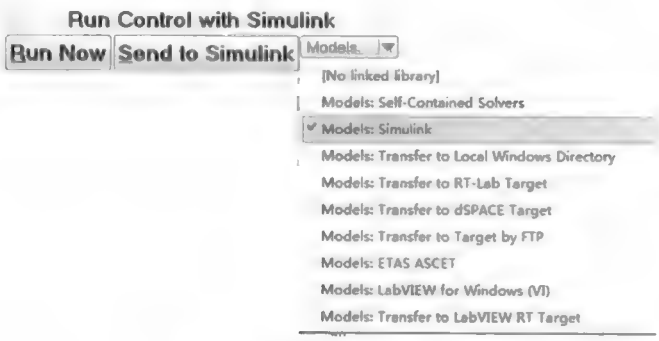


图 4.21 选择 Simulink 接口

ⓑ 点击如图 4.22 所示的选项，选择“[Link to New Dataset]”

ⓒ 此时将弹出一个如图 4.23 所示的对话框，在对话框中依次输入“Example”和“MPCtest1”，点击“Create and Link”完成新建，如图 4.24 所示。

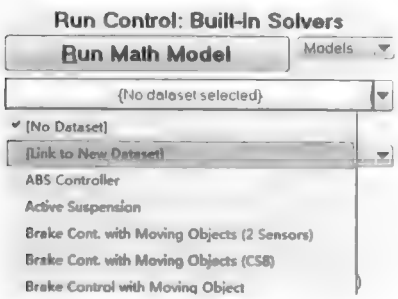


图 4.22 Link to New Dataset

图 4.23 新建一个与 Simulink 联合的 Dataset

ⓓ 点击如图 4.25 所示的“MPCtest1”，将会弹出如图 4.26 所示的 Simulink [Example] MPCtest1 的主界面，如图 4.27所示，选择相关路径。

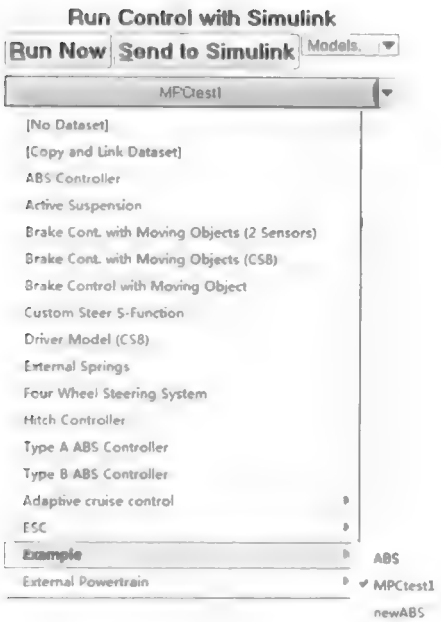


图 4.24　选择新建的“MPCtest1” dataset

➢ 浏览工作路径：D：\ Users \ Public \ Documents \ CarSim_ Data。

➢ Simulink Model 的路径：D：\ Users \ Public \ Documents \ CarSim_ Data \ Extensions \ Simulink \ MPCtest1. mdl。

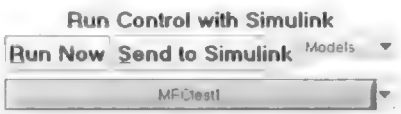


图 4.25　点击“MPCtest1”

目录下新建一个空白的 Simulink Model，命名为 MPCtest1. mdl，点击“浏览 MPCtest1. mdl，如图 4.27 所示。

(e) 定义 CarSim 的导入变量

点击如图 4.28（a）所示的“Import Channels”，选择“I/O Channels：Import”；然后点击如图 4.28（b）所示的“[Copy and Link Dataset]”；如图 4.28（c）所示，输入“MPCtest1 input”。

点击“MPCtest1 input”图标，显示如图 4.29 所示界面，这里需要浏览找到“Readme file for imports”图标，其后的文本框里显示“Programs \ solvers \ ReadMe \ f_ i_ i_ s_ imports_ tab. txt”，定义 CarSim 的导入变量为车速和 4 个车轮的转角，顺序依次为 IMP_ SPEED（质心车速，km/h），IMP_ STEER_ L1（左前轮转角，°），IMP_ STEER_ R1（右前轮转角，°），IMP_ STEER_ L2（左后轮转角，°），以及 IMP_ STEER_ R2（右后轮转角，°），如图 4.30 所示。

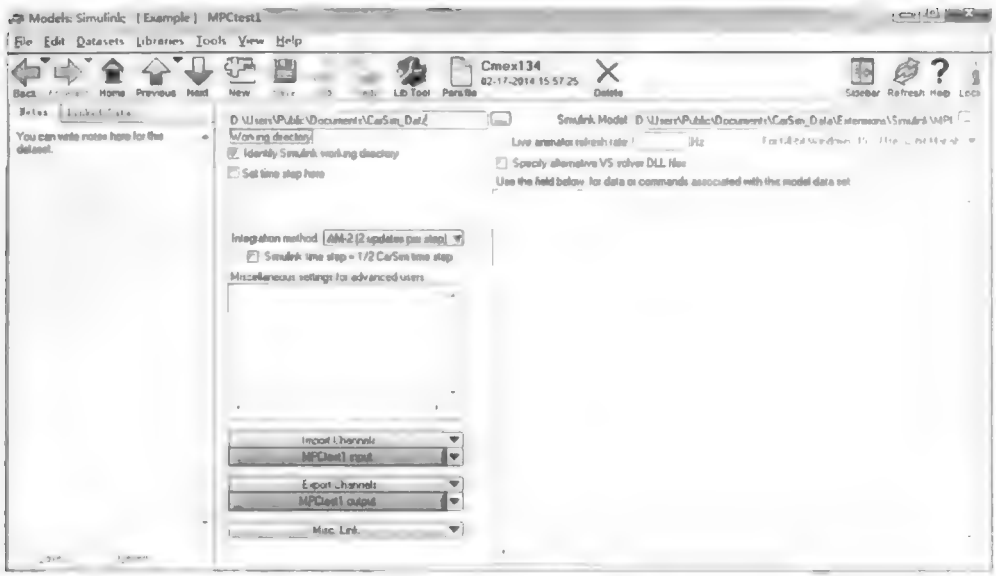


图 4.26　Simulink｛Example｝ MPCtest1 的主界面

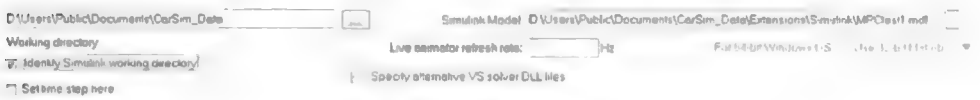
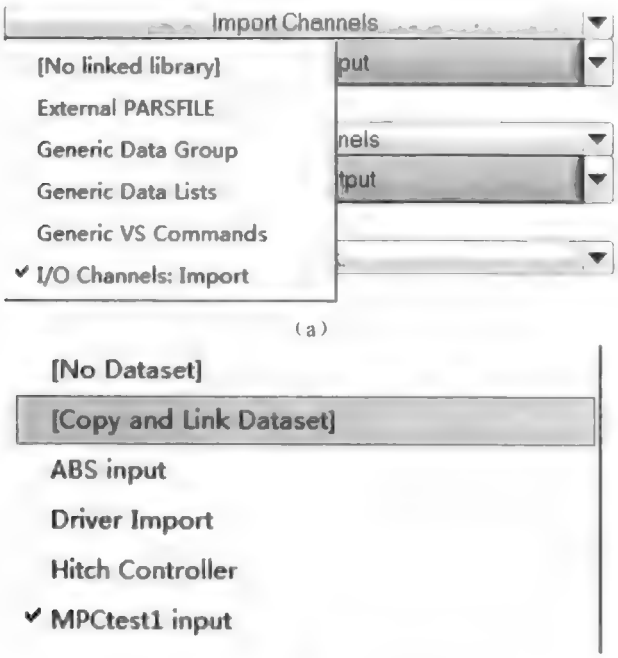


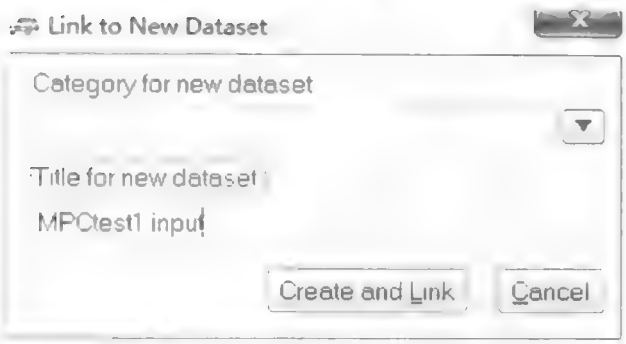
图 4.27　选择路径



（a）

（b）

图 4.28　新建 MPCtest1 input



（c）

图 4.28　新建 MPCtest1 input（续）

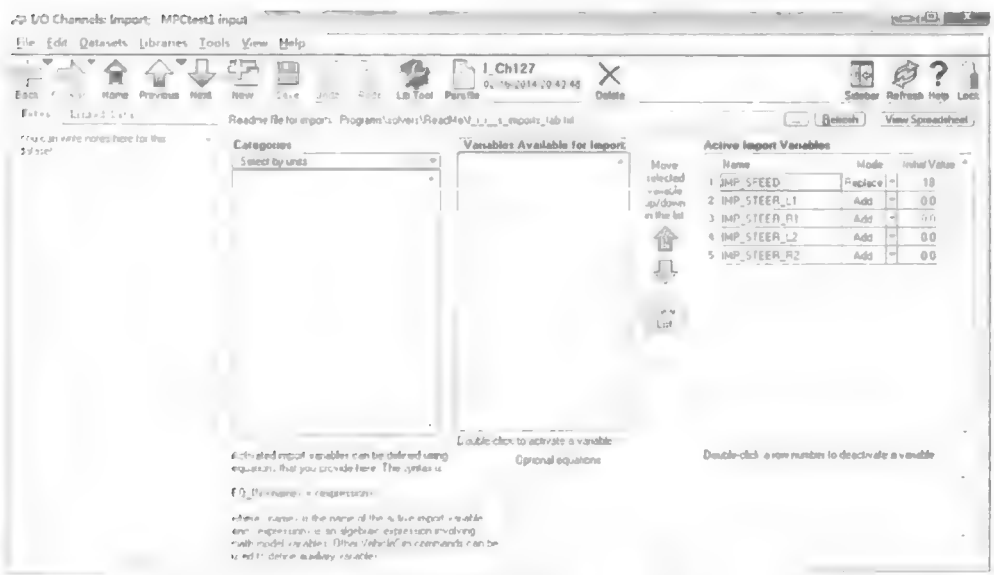


图 4.29　定义 CarSim 导入变量

需要注意的是：CarSim 的导入变量和 Simulink 中 MPC 模型的输出量是相对应的，所以这里的 CarSim 的导入变量应按照如图 4.30 的顺序排列选择。

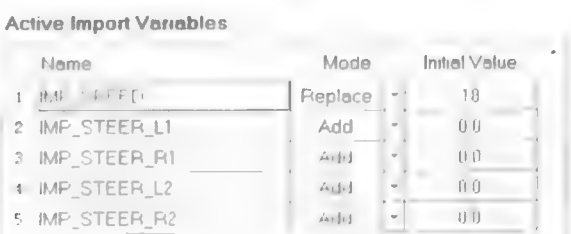


图 4.30　CarSim 的导入变量

ⓕ 定义 CarSim 的导出变量。

步骤同ⓔ定义 CarSim 导入变量相同，首先新建名为“MPCtest1 output”的一个 dataset。点击“MPCtest1 output”图标，弹出如图 4.31 所示的界面这里需要浏览找到“Readme file for outputs”图标，其后的文本框内显示为“Programs \ solvers \ ReadMe \ f_ i_ outputs_ tab. txt” 定义 CarSim 的导出变量依次为 X0（坐标系 X 轴的坐标值［m］），Y0（坐标系 Y 轴的坐标值［m］），Yaw（偏航角［°］），Vx（质心处的纵向车速［km/h］），以及 Steer_ SW（方向盘转角［°］）。

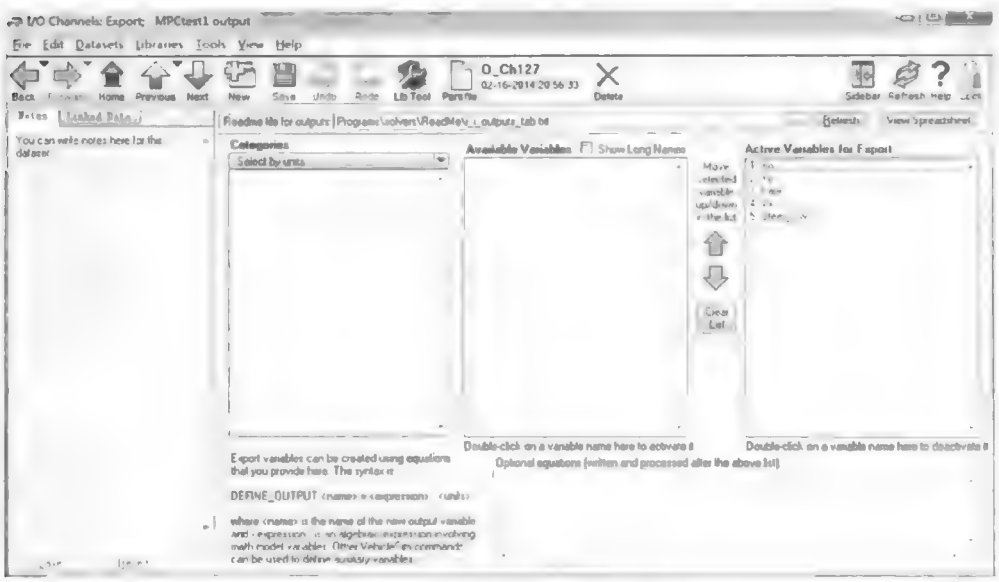


图 4.31 定义 CarSim 导出变量

需要注意的是：CarSim 的导出变量（X0、Y0 和 Yaw）和 Simulink 中 MPC 模型的输入量是相对应的，CarSim 的导出变量（Vx 和 Steer_ SW）为观测量，所以这里的 CarSim 的导出变量应按照如图 4.32 的顺序排列选择

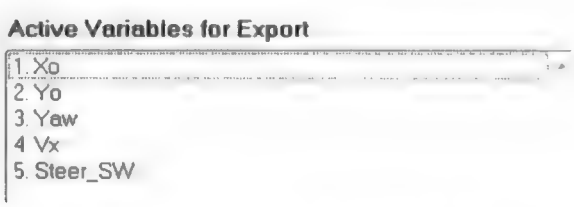


图 4.32 CarSim 的导出变量

ⓖ 点击“Home”图标，返回 CarSim 的主界面，点击如图 4.33 所示的“Send to Simulink”图标，此时 Matlab 及之前新建的空白模型 MPCtest1.mdl 将被打开，如图 4.34所示。

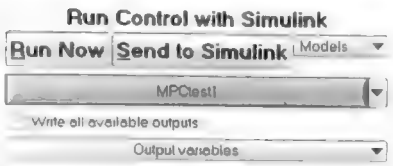


图 4.33　数学模型求解器

h 如图 4.35 所示，在“Matlab Command Window”中输入“Simulink”，即可打开 Simulink 库浏览窗口。注意现在的 Simulink 库浏览窗口比单独运算 Matlab/Simulink 时多了一个“CarSim S - Function”图标。如图 4.36 所示，将“CarSim S - Function”图标拖曳到 MPCtest1.mdl 中，生成的新模块恰好有一个输入接口和一个输出接口，分别对应着 CarSim 的导入变量和导出变量。

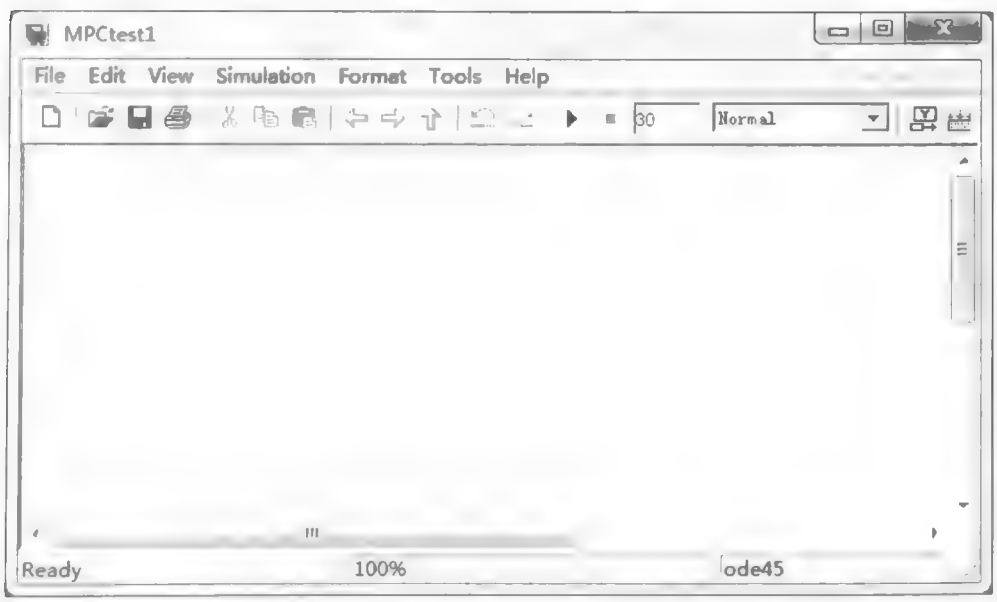


图 4.34　MPCtest1.mdl

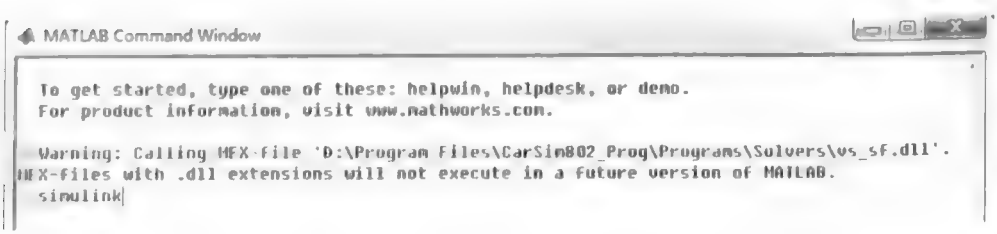


图 4.35　Matlab Command Window 命令窗口

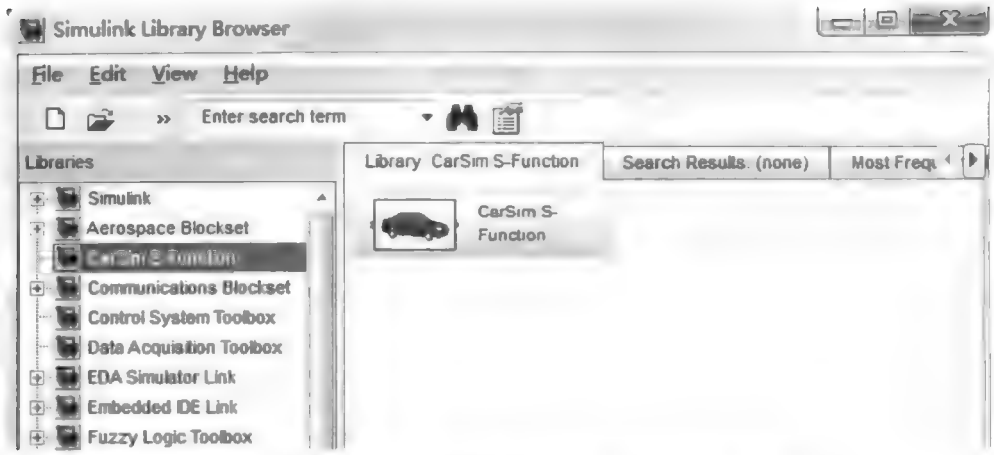


图 4.36　含有 CarSim *S* 函数模块的 Simulink Library Browser

完成上述设定后，CarSim 通过外部接口将车辆模型发送至指定路径下的 Simulink 仿真文件中，CarSim 模块即以 *S* 函数的形式增加到 Simulink 模型库中

ⓘ 在 MPCtest1. mdl 中加入基于 MPC 的轨迹跟踪控制器，CarSim 的导出量经过该控制器的计算，决策出下一时刻的质心车速和前轮偏角，然后导入到 CarSim 模块里。

将 Simulink 浏览窗口中的“S - Function”图标拖曳到 MPCtest1. mdl 中，显示新的 *S* 函数模块。双击该图标，弹出如图 4.37 所示的对话框，在“S -

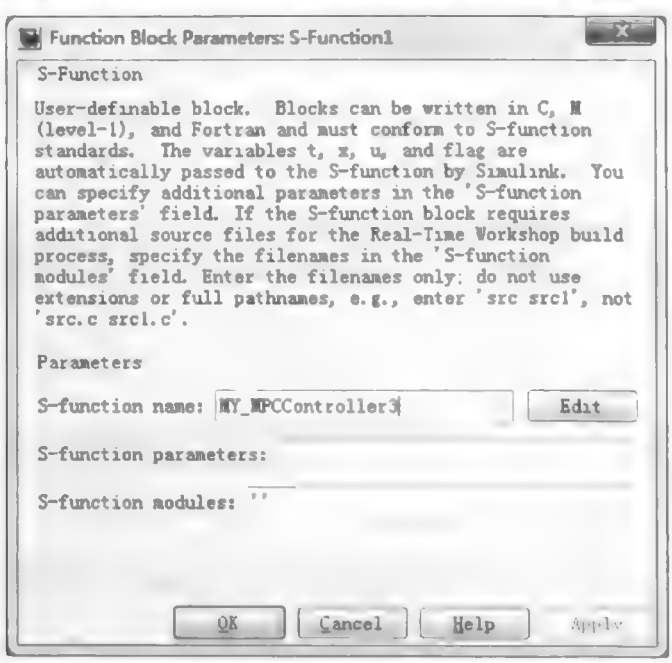


图 4.37　S 函数对话框

Function name”中输入“MY_ MPCController3”，点击“Edit”，将设计好的基于 MPC 的轨迹跟踪控制算法的 m 文件导入，显示为“MY_ MPCController3”的 S 函数图标。注意此模块恰好有一个输入接口和一个输出接口，分别对应着轨迹跟踪控制器（MY_ MPCControllers）的导入变量和导出变量。

其中，MY_ MPCController3 控制器模块的输入量为上一时刻 CarSim 模块的导出变量 X0，Y0 和 Yaw。在如图 4.38 所示的 Simulink/CarSim 联合仿真平台中加入延时模块。

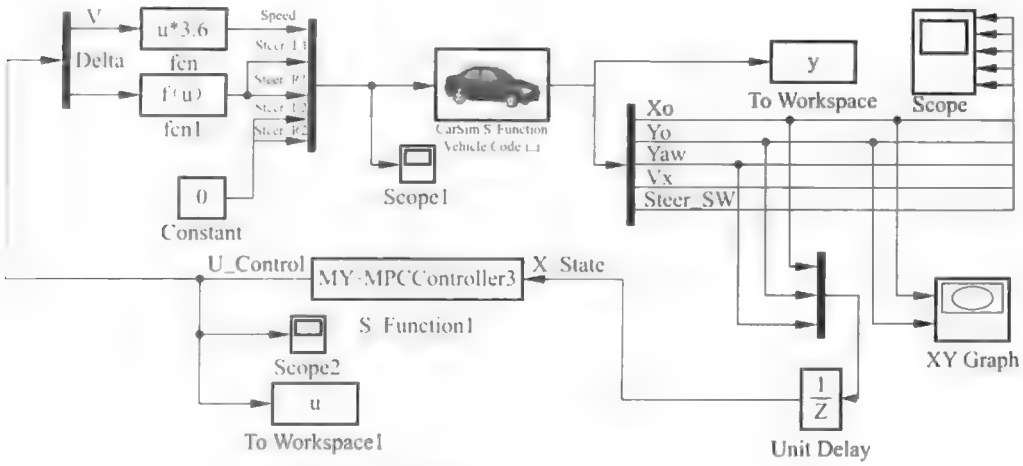


图 4.38　Simulink/CarSim 联合仿真平台

MY_ MPCController3 控制器模块的输出量为车速 v（m/s）和前轮偏角 δ（deg），需对其转换成满足 CarSim 模块要求的导入变量：IMP_ SPEED（质心车速［km/h］），IMP_ STEER_ L1（左前轮转角［°］），IMP_ STEER_ R1（右前轮转角［°］），IMP_ STEER_ L2（左后轮转角［°］），以及 IMP_ STEER_ R2（右后轮转角［°］）。所以，在 MY_ MPCController3 控制器模块的输出量与 CarSim 模块的导入变量之间加入了转换函数。

定义：

$$\begin{bmatrix} IMP\_SPEED \\ IMP\_STEER\_L1 \\ IMP\_STEER\_R1 \\ IMP\_STEER\_L2 \\ IMP\_STEER\_R2 \end{bmatrix} = \begin{bmatrix} 3.6 & 0 \\ 0 & 180/\mathrm{pi} \\ 0 & 180/\mathrm{pi} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v \\ \delta \end{bmatrix} \tag{4.24}$$

j 返回 CarSim 主界面，点击“Run Now”运行结束后，再点击“Send to Simulink”，弹出 MPCtest1. mdl，如图 4.38 所示，运行整个模型。

说明：若对 CarSim 软件的参数（车辆参数、仿真工况等）进行修改，则要再点击“Run Now”，运行结束后，再点击“Send to Simulink”，来重新读入修改后的参数，最后运行模型。若只在 Simulink 模型里做相应的修改，不会影响 CarSim 中的参数，直接点击模型。在 Simulink 中运行仿真模型，跟踪圆形，可观测到轨迹曲线，如图 4.39 所示。

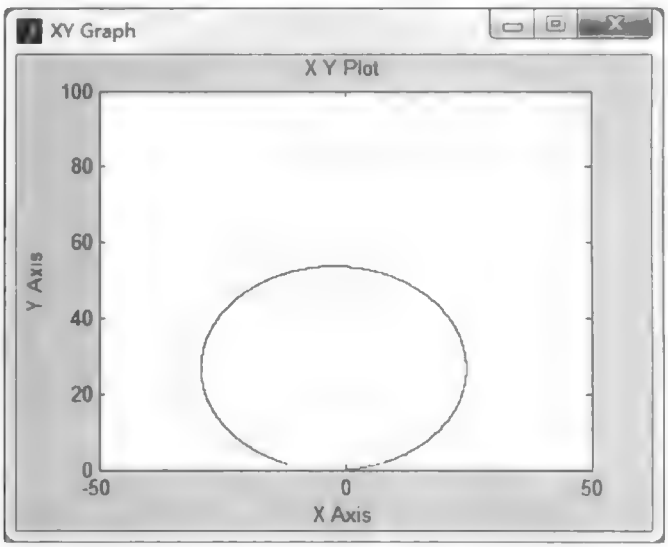


图 4.39　Simulink 中加入控制器后的轨迹曲线

ⓚ 观察加入基于 MPC 的轨迹跟踪控制器后车辆跟踪圆形轨迹，点击“Animate”，查看动画，如图 4.40 所示；同样可点击“Plot”绘制曲线

图 4.40　仿真动画

4.4.2　基于 MPC 的轨迹跟踪控制器的设计

本节主要介绍如何通过 Simulink 的 S 函数设计基于 MPC 的轨迹跟踪控制器。S 函数为 Simulink 中的一个模块，其运算过程符合 Simulink 模块的仿真过程。S 函数格式非常严格，在 Simulink 中有一个模板 m 文件，使用时可以在此模板上简单修改。此模板 m 文件由主函数以及子函数（不同 flag 值调用的函数）组成。该 m 文件可直接调用生成 MY_ MPCController3 控制器的 S 函数。该程序详见 chapter4_ 4_ 3. m 文件。

为便于代码解析，将仿真程序分为若干模块分别介绍，程序后面有备注，方便读者阅读。

第一部分为主函数：

```
function[sys,x0,str,ts]=MY_MPCController3(t,x,u,flag)
switch flag,
case 0  % 初始化
[sys,x0,str,ts]=mdlInitializeSizes;  % Initialization
case 2  % 更新离散状态量
sys=mdlUpdates(t,x,u);  % Update discrete states
case 3  % 计算输出
sys=mdlOutputs(t,x,u);% Calculate outputs
case {1,4,9}  % Unused flags
sys=[];
otherwise% 未知的 flag 值
error(['Unhandled flag=',num2str(flag)]);  % Error handling
end
% S 函数主程序结束
```

第二部分为初始化子函数：

```
function[sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates  =0;  % 连续状态量的个数
sizes.NumDiscStates  =3;  % 离散状态量的个数
sizes.NumOutputs     =2;  % 输出量的个数
```

```
sizes.NumInputs        =3;  % 输入量的个数
sizes.DirFeedthrough =1;  % Matrix D is non-empty. 直接贯
                            通标示
sizes.NumSampleTimes =1;     % 采样时间的个数
sys =simsizes(sizes);
x0 =[0;0;0];  % 状态量初始化
global U;
U =[0;0];
str =[];              % Set str to an empty matrix.
ts   =[0.05 0];      % sample time:[period,offset]
% 初始化子函数结束
```

第三部分为更新离散状态量子函数：

```
function sys =mdlUpdates(t,x,u)
sys =x;
% End of mdlUpdate.
```

第四部分为计算输出子函数，是轨迹跟踪控制的主体，在程序解析时我们会将第二节的理论算法与代码对应，方便读者的理解

```
function sys =mdlOutputs(t,x,u)
global a b u_piao;
global U;
global kesi;
Nx =3;     % 状态量的个数
Nu =2;     % 控制量的个数
Np =60;    % 预测步长
Nc =30;    % 控制步长
Row =10;   % 松弛因子
fprintf('Update start,t =% 6.3f\n',t)
t_d =u(3) *3.1415926/180;% CarSim输出的为角度，角度转换为弧度
% 期望轨迹为圆形轨迹的代码，读者可根据实际情况设计所需路径方程，
% 源代码文件中有直线轨迹代码
% 半径为25 m的圆形轨迹，速度为5m/s
```

```
r(1)=25*sin(0.2*t);
r(2)=25+10-25*cos(0.2*t);
r(3)=0.2*t;
vd1=5;
vd2=0.104;
%     % 半径为25m的圆形轨迹,速度为3m/s
%     r(1)=25*sin(0.12*t);
%     r(2)=25+10-25*cos(0.12*t);
%     r(3)=0.12*t;
%     vd1=3;
%     vd2=0.104;
%     半径为25m的圆形轨迹,速度为10m/s
%       r(1)=25*sin(0.4*t);
%       r(2)=25+10-25*cos(0.4*t);
%       r(3)=0.4*t;
%       vd1=10;
%       vd2=0.104;
%       % 半径为25m的圆形轨迹,速度为4m/s
%       r(1)=25*sin(0.16*t);
%       r(2)=25+10-25*cos(0.16*t);
%       r(3)=0.16*t;
%       vd1=4;
%       vd2=0.104;
%
% 参数设计
%
kesi=zeros(Nx+Nu,1);
kesi(1)=u(1)-r(1);  % u(1)==X(1)
kesi(2)=u(2)-r(2);  % u(2)==X(2)
kesi(3)=t_d-r(3);   % u(3)==X(3)
kesi(4)=U(1);
kesi(5)=U(2);
fprintf('Update start,u(1)=%4.2f\n',U(1))
fprintf('Update start,u(2)=%4.2f\n',U(2))
```

```
T=0.05;
T_all=40;  % 总的仿真时间,主要功能是防止计算期望轨迹越界
% Mobile Robot/ Vehicle Parameters
L=2.6;
% Mobile Robot/ Vehicle variable
% ================================================================
% 矩阵初始化
% ================================================================
u_piao=zeros(Nx,Nu);
Q=eye(Nx*Np,Nx*Np);
R=5*eye(Nu*Nc);
a=[1     0    -vd1*sin(t_d)*T;
0     1   vd1*cos(t_d)*T;
0     0   1;];
b=[cos(t_d)*T   0;
sin(t_d)*T   0;
tan(vd2)*T/L        vd1*T/(cos(vd2)^2);];
% 以上代码对应式(4.6)中的参数
A_cell=cell(2,2);
B_cell=cell(2,1);
A_cell{1,1}=a;
A_cell{1,2}=b;
A_cell{2,1}=zeros(Nu,Nx);
A_cell{2,2}=eye(Nu);
B_cell{1,1}=b;
B_cell{2,1}=eye(Nu);
A=cell2mat(A_cell);
B=cell2mat(B_cell);
C=[1 0 0 0 0;0 1 0 0 0;0 0 1 0 0;];
% 以上对应式(4.10)中的参数
PHI_cell=cell(Np,1);
THETA_cell=cell(Np,Nc);
for j=1:1:Np
        PHI_cell{j,1}=C*A^j;
```

```
        for k = 1:1:Nc
            if k <= j
                THETA_cell{j,k} = C * A^(j-k) * B;
            else
                THETA_cell{j,k} = zeros(Nx,Nu);
            end
        end
    end
    PHI = cell2mat(PHI_cell);        % size(PHI)=[Nx * Np  Nx+Nu]
    THETA = cell2mat(THETA_cell);    % size(THETA)=[Nx * Np  Nu *
                                                        (Nc+1)]
    % 以上对应式(4.12)中的参数
    H_cell = cell(2,2);
    H_cell{1,1} = THETA' * Q * THETA + R;
    H_cell{1,2} = zeros(Nu * Nc,1);
    H_cell{2,1} = zeros(1,Nu * Nc);
    H_cell{2,2} = Row;
    H = cell2mat(H_cell);
    error = PHI * kesi;
    f_cell = cell(1,2);
    f_cell{1,1} = 2 * error' * Q * THETA;
    f_cell{1,2} = 0;
    f = cell2mat(f_cell);
    % 以上对应式(4.19)中的参数
    %% ------------------------------------------------------------------
    % 不等式约束
    %% ------------------------------------------------------------------
    A_t = zeros(Nc,Nc);
    for p = 1:1:Nc
        for q = 1:1:Nc
            if q <= p
                A_t(p,q) = 1;
            else
                A_t(p,q) = 0;
```

```
        end
    end
end
A_I = kron(A_t,eye(Nu));% 对应式(4.17)中的参数
Ut = kron(ones(Nc,1),U);
umin =[ -0.2; -0.54;];% 维数与控制变量的个数相同
umax =[0.2;0.332];
delta_umin =[ -0.05; -0.0082;];
delta_umax =[0.05;0.0082];
Umin = kron(ones(Nc,1),umin);
Umax = kron(ones(Nc,1),umax);
A_cons_cell= {A_I zeros(Nu * Nc,1); -A_I zeros(Nu * Nc,1)};
b_cons_cell={Umax - Ut; -Umin + Ut};
A_cons = cell2mat(A_cons_cell);
% (求解方程)状态量不等式约束增益矩阵,转换为绝对值的取值范围
b_cons = cell2mat(b_cons_cell);
% (求解方程)状态量不等式约束的取值
% ==========================================================
% 状态量约束
% ==========================================================
M=10;
delta_Umin = kron(ones(Nc,1),delta_umin);
delta_Umax = kron(ones(Nc,1),delta_umax);
lb =[delta_Umin;0];%(求解方程)状态量下界,包含控制时域内控制增
  量和松弛因子
ub =[delta_Umax;M];%(求解方程)状态量上界,包含控制时域内控制增
    量和松弛因子
% ==========================================================
% 开始求解过程
% ==========================================================
options = optimset('Algorithm','active - set');
% options = optimset('Algorithm','interior - point - convex');
[X,fval,exitflag]=quadprog(H,f,A_cons,b_cons,[],[],lb,
   ub,[],options);
```

```
% ------------------------------------------------------------------
% 计算输出
% ------------------------------------------------------------------
u_piao(1)=X(1);
u_piao(2)=X(2);
U(1)=kesi(4)+u_piao(1);% 用于存储上一个时刻的控制量
U(2)=kesi(5)+u_piao(2);
u_real(1)=U(1)+vd1;
u_real(2)=U(2)+vd2;
sys=u_real;
toc
% End of mdlOutputs.
```

4.5　基于运动学模型的轨迹跟踪仿真结果分析

为了验证所设计的轨迹跟踪控制器，在不同的轨迹下对其跟踪能力进行仿真测试。基于运动学模型的轨迹跟踪控制器主要用于泊车等低速工况，根据其运动特点，选取直线轨迹和圆形轨迹进行跟踪。

（1）直接轨迹跟踪

分别在参考速度为3 m/s、5 m/s和10 m/s，参考前轮偏角为0的情况下进行仿真测试，直线轨迹以参数方程的形式给出：

$$\begin{cases} x(t) = v_d t \\ y(t) = 5 \\ \varphi(t) = 0 \end{cases} \tag{4.25}$$

式中，v_d为期望的纵向速度。

设置初始时刻为$t_0=0$，末端时刻$t_{final}=50$ s，初始状态设置为：$x_0=x(t_0)=0$，$y_0=y(t_0)=0$，$\varphi_0=\varphi(t_0)=0$。约束条件采用控制量约束和控制增量约束，设置与式（4.18）保持一致。模型预测控制器的基本参数设置为：预测时域$N_p=60$，控制时域$N_c=30$，控制周期$T=0.05$ s。

仿真结果如图4.41、图4.42和图4.43所示。其中，图4.41（a）为车辆跟踪直线轨迹的结果，图4.41（b）为对应的轨迹跟踪偏差。从图4.41中可以看出，在具备初始偏差的情况下，3种不同参考速度下无人驾驶车辆都能迅速跟踪直线轨迹。进一步观察发现，10 m/s的情况下出现轻微的超调。这主要是因为在该速度下使用设定的预测时域已经不太适用。通过增大

预测时域的范围，可以消除这种现象。也就是说，随着速度的增大，预测时域的范围要求越大。图 4.42 和图 4.43 为控制量（前轮偏角和速度）随时间的变化曲线。可以看出，无论是前轮偏角，还是速度，都能快速跟踪期望轨迹并被限定在约束范围内。通过右侧的局部放大图能够进一步观察到跟踪过程中控制增量没有突变的现象，每个控制周期内的控制增量都保持在约束范围内。

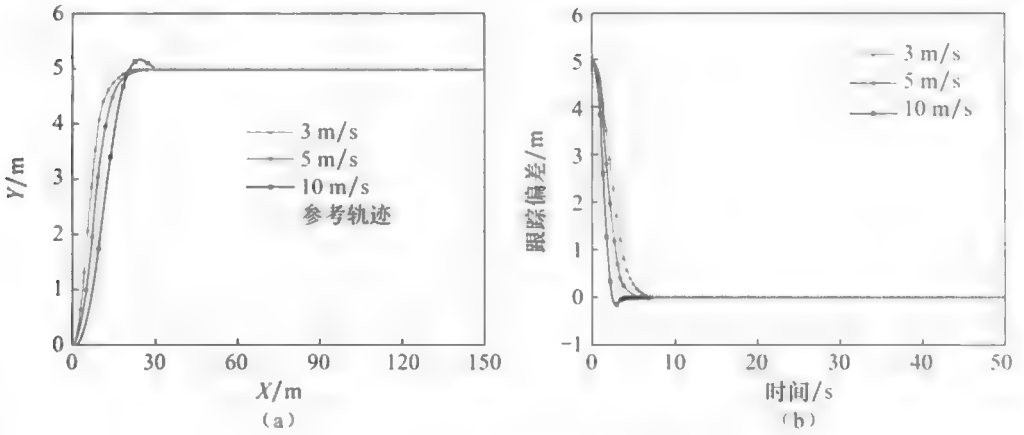


图 4.41　跟踪直线轨迹结果与跟踪偏差

（a）参考轨迹与实际轨迹；（b）跟踪偏差

（2）圆形轨迹跟踪

圆形参考轨迹依然以参数方程的形式给出：

$$\begin{cases} x(t) = 25\sin\varphi_t \\ y(t) = 35 - 25\cos\varphi_t \\ \varphi(t) = \dfrac{v_d \tan\delta_d}{l} t \end{cases} \tag{4.26}$$

式中，δ_d 为期望前轮偏角。

仿真过程中 v_d 依次取为 3 m/s，5 m/s 和 10 m/s，δ_d 设定为 5.94°。设置初始时刻为 $t_0 = 0$，末端时刻 $t_{final} = 50$ s，初始状态设置为：$x_0 = x(t_0) = 0$，$y_0 = y(t_0) = 0$。约束条件和模型预测控制器的基本参数设置与直线轨迹仿真保持一致。

仿真结果如图 4.44、图 4.45 和图 4.46 所示。其中，图 4.44（a）为车辆跟踪圆形轨迹时位置的变化情况，图 4.44（b）为轨迹跟踪偏差。从图 4.44

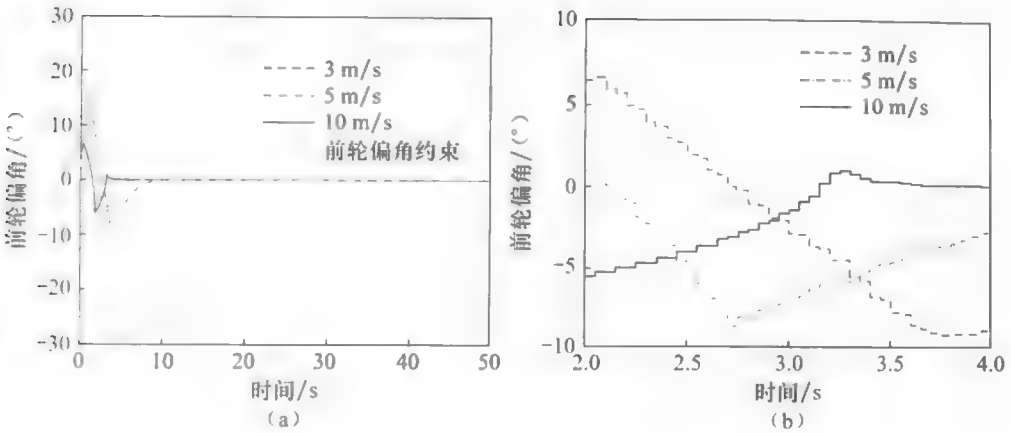


图 4.42　跟踪直线轨迹的前轮偏角

（a）前轮偏角随时间变化；（b）前轮偏角局部放大

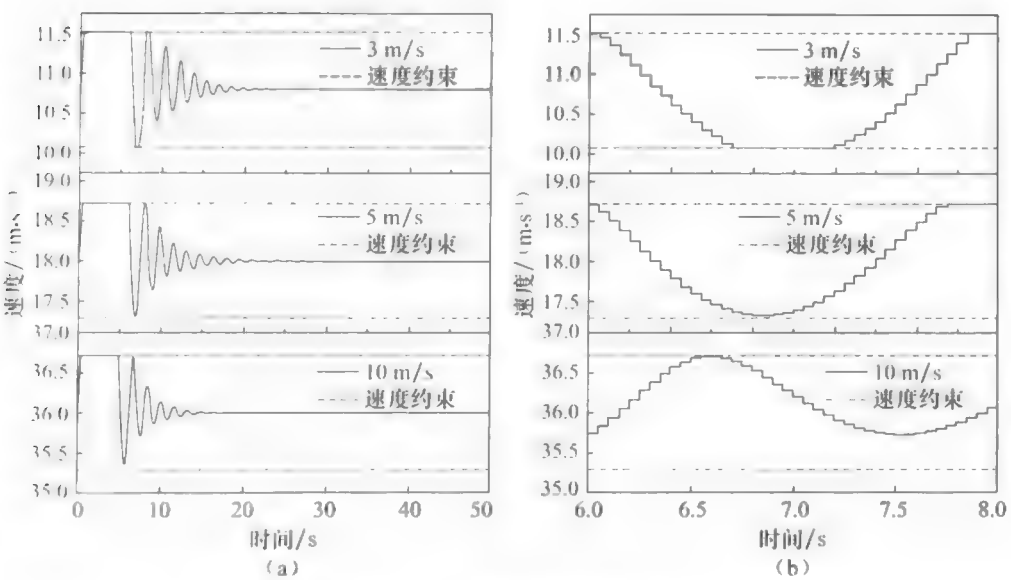


图 4.43　跟踪直线轨迹的速度

（a）速度随时间变化关系；（b）速度局部放大图

中可以看出，无人驾驶车辆在控制器的作用下能快速跟踪期望轨迹，系统跟踪误差最终收敛为0，具有良好的全局稳定性。图 4.45、图 4.46 为跟踪过程中控制量（前轮偏角和纵向车速）随时间的变化。在整个跟踪过程中，控制量

始终保持在给定的约束条件范围内。从右侧的局部放大图可以进一步看出，控制器所给出的控制增量也被限定在了设定的范围内，没有出现任何突变的现象。因此，控制器不仅实现了快速跟踪期望轨迹的功能，也保证了跟踪过程的平稳性。

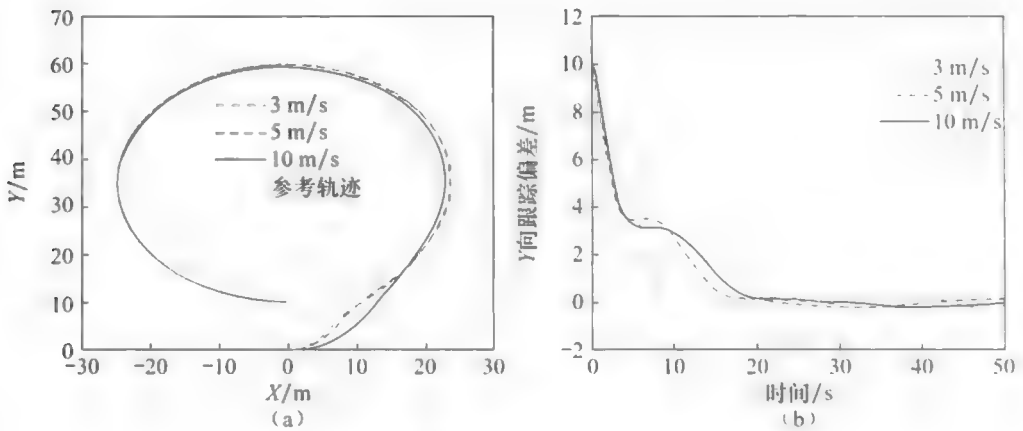


图 4.44　跟踪圆形轨迹结果与跟踪偏差

(a) 参考轨迹与实际轨迹；(b) 跟踪偏差

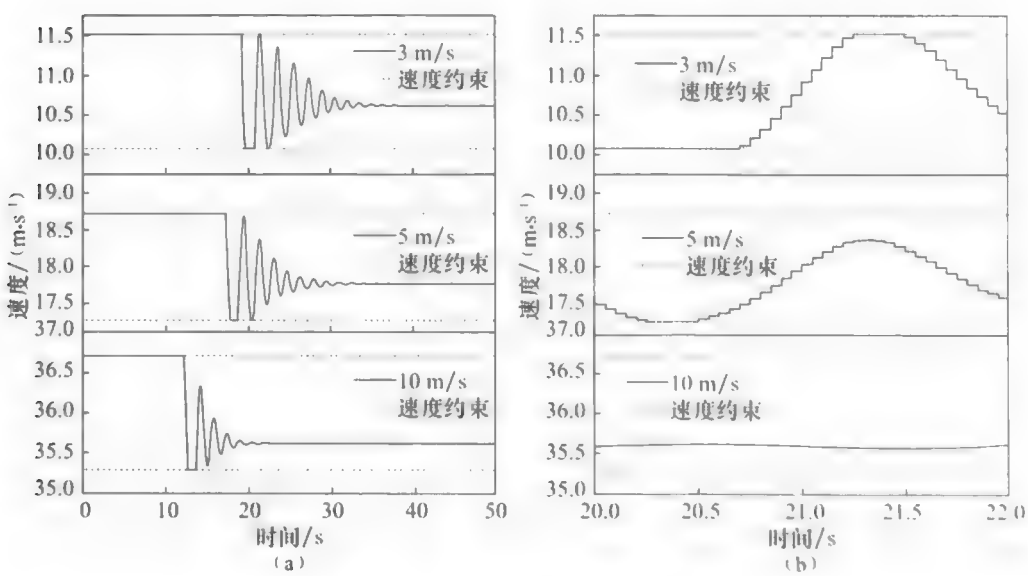


图 4.45　跟踪圆形轨迹的速度

(a) 纵向速度随时间变化；(b) 纵向速度局部放大

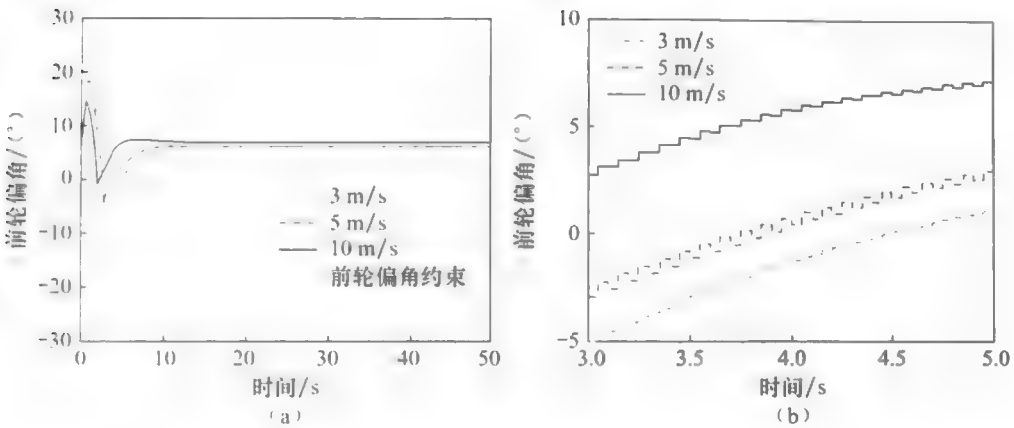


图 4.46　跟踪圆形轨迹的前轮偏角

(a) 前轮偏角随时间变化；(b) 前轮偏角局部放大

第5章

基于动力学模型的无人驾驶车辆主动转向控制

上一章详细介绍了基于车辆运动学模型的无人驾驶车辆轨迹跟踪方法，并通过搭建 Simulink/CarSim 联合仿真平台对算法进行了验证，初步掌握了利用 S 函数编写模型预测控制器以及在 CarSim 中搭建车辆模型的方法；但仅仅掌握这些知识还不足以将模型预测的方法真正应用在无人驾驶车辆上。

无人驾驶车辆不同于一般的室内轮式机器人，它更多是以较高速度在复杂的交通环境中行驶。为了提高无人驾驶车辆在高速行驶时的可靠性，有必要在控制器中引入车辆动力学模型，以准确的动力学模型作为预测模型，可以提高控制器对车辆未来行为的预估能力，进而在保证车辆稳定运行的同时，充分利用车辆的潜能；同时，无人驾驶车辆在高速运动下执行机构的控制输入，轮胎与地面摩擦引起的滑移，以及横向加速度引起的侧倾等动力学非线性约束条件比低速下更加苛刻。对这些约束进行深入分析将进一步保障车辆行驶的安全性与稳定性。

本章将以无人驾驶车辆的主动转向控制为应用背景，建立以车辆动力学模型为基础的模型预测控制器。结合车辆高速运动下的各类约束，进一步说明复杂约束条件下模型预测控制的求解方法。最后，依然以 Simulink/CarSim 联合仿真平台验证算法的有效性。

5.1 理论基础

5.1.1 线性误差方程

考虑式（2.38）所建立的非线性动力学模型，$\dot{\boldsymbol{\xi}}_{\mathrm{dyn}}=f_{\mathrm{dyn}}(\dot{\boldsymbol{\xi}}_{\mathrm{dyn}},\ \boldsymbol{u}_{\mathrm{dyn}})$。其中，状态量为$\boldsymbol{\xi}_{\mathrm{dyn}}=[\dot{y},\ \dot{x},\ \varphi,\ \dot{\varphi},\ Y,\ X]^{\mathrm{T}}$ 控制量选取为$\boldsymbol{u}_{\mathrm{dyn}}=\delta_{\mathrm{f}}$，即只对车辆在轨迹跟踪过程中的前轮偏角进行控制，而车辆纵向速度保持不变。无人驾驶车辆在高速行驶时对控制器实时性要求更加严格，非线性模型预测控制难以满足 本章继续采用线性时变模型预测控制，因此首先需要对非线性动力学模型进行线性化

由于期望轨迹无法给出所有状态点的信息，因此采用第3章中的方式B对其进行线性化，得到线性时变方程为：

$$\dot{\boldsymbol{\xi}}_{\mathrm{dyn}}=\boldsymbol{A}_{\mathrm{dyn}}(t)\boldsymbol{\xi}_{\mathrm{dyn}}(t)+\boldsymbol{B}_{\mathrm{dyn}}(t)\boldsymbol{u}_{\mathrm{dyn}}(t) \tag{5.1}$$

式中，$\boldsymbol{B}_{\mathrm{dyn}}(t)=\left.\dfrac{\partial f_{\mathrm{dyn}}}{\partial \boldsymbol{u}_{\mathrm{dyn}}}\right|_{\boldsymbol{\xi}_t,\boldsymbol{u}_t}=\left[\dfrac{2C_{\mathrm{cf}}}{m},\ \dfrac{2C_{\mathrm{cf}}\left(2\delta_{\mathrm{f},t-1}-\dfrac{\dot{y}_t+a\dot{\varphi}_t}{\dot{x}_t}\right)}{m},\ 0,\ \dfrac{2aC_{\mathrm{cf}}}{I_z},\ 0,\ 0\right]$，

$$\boldsymbol{A}_{\mathrm{dyn}}(t)=\left.\frac{\partial f_{\mathrm{dyn}}}{\partial \boldsymbol{\xi}_{\mathrm{dyn}}}\right|_{\boldsymbol{\xi}_t,\boldsymbol{u}_t}$$

$$=\begin{bmatrix}
\dfrac{-2(C_{\mathrm{cf}}+C_{\mathrm{cr}})}{m\dot{x}_t} & \dfrac{\partial f_{\dot{y}}}{\partial \dot{x}} & 0 & -\dot{x}_t+\dfrac{2(bC_{\mathrm{cr}}-aC_{\mathrm{cf}})}{m\dot{x}_t} & 0 & 0\\
\dot{\varphi}-\dfrac{2C_{\mathrm{cf}}\delta_{\mathrm{f},t-1}}{m\dot{x}_t} & \dfrac{\partial f_{\dot{x}}}{\partial \dot{x}} & 0 & \dot{y}_t-\dfrac{2aC_{\mathrm{cf}}\delta_{\mathrm{f},t-1}}{m\dot{x}_t} & 0 & 0\\
0 & 0 & 0 & 1 & 0 & 0\\
\dfrac{2(bC_{\mathrm{cr}}-aC_{\mathrm{cf}})}{I_z\dot{x}_t} & \dfrac{\partial f_{\dot{\varphi}}}{\partial \dot{x}} & 0 & \dfrac{-2(a^2C_{\mathrm{cf}}+b^2C_{\mathrm{cr}})}{I_z\dot{x}_t} & 0 & 0\\
\cos(\varphi_t) & \sin(\varphi_t) & \dot{x}_t\cos(\varphi_t)-\dot{y}_t\sin(\varphi_t) & 0 & 0 & 0\\
-\sin(\varphi_t) & \cos(\varphi_t) & -\dot{y}_t\cos(\varphi_t)-\dot{x}_t\sin(\varphi_t) & 0 & 0 & 0
\end{bmatrix}$$

其中，$\dfrac{\partial f_{\dot{y}}}{\partial \dot{x}}=(2C_{\mathrm{cf}}(\dot{y}_t+a\dot{\varphi}_t)+2C_{\mathrm{cr}}(\dot{y}_t-b\dot{\varphi}_t))/m\dot{x}_t^2-\dot{\varphi}_t$，$\dfrac{\partial f_{\dot{x}}}{\partial \dot{x}}=(2C_{\mathrm{cf}}\delta_{\mathrm{f},t-1}(\dot{y}_t+a\dot{\varphi}_t))/(m\dot{x}_t^2)$，$\dfrac{\partial f_{\dot{\varphi}}}{\partial \dot{x}}=(2aC_{\mathrm{cf}}(\dot{y}_t+a\dot{\varphi}_t)-2bC_{\mathrm{cr}}(\dot{y}_t-b\dot{\varphi}_t))/I_z\dot{x}_t^2$

同样，对式（5.1）采用一阶差商的方法进行离散化处理，得到离散的状态空间表达式：

$$\boldsymbol{\xi}_{\mathrm{dyn}}(k+1)=\boldsymbol{A}_{\mathrm{dyn}}(k)\boldsymbol{\xi}_{\mathrm{dyn}}(k)+\boldsymbol{B}_{\mathrm{dyn}}(k)\boldsymbol{u}_{\mathrm{dyn}}(k) \tag{5.2}$$

式中，$\boldsymbol{A}_{\mathrm{dyn}}(k)=\boldsymbol{I}+T\boldsymbol{A}_{\mathrm{dyn}}(t)$，$\boldsymbol{B}_{\mathrm{dyn}}(k)=T\boldsymbol{B}_{\mathrm{dyn}}(t)$。

为方便读者对不同的车辆动力学模型进行线性化处理，以下提供本书在处理过程中所用的 Matlab 代码，程序为 chapter_ 5_ 1_ 1. m 读者只要根据建立的车辆动力学模型修改程序中的车辆固有参数和微分方程，就可以得到最终结果。

```
clc
clear all;
%% 以下为程序主体
% 车辆参数定义
syms x_dot y_dot phi phi_dot Y X;% 车辆状态量
syms delta_f  % 控制量(前轮偏角)
% 车辆前、后轮滑移率
Sf=0.2;Sr=0.2;
% 前、后轮距离车辆质心的距离,车辆固有参数
a=1.232;b=1.468;
% 前、后轮的纵横向侧偏刚度,车辆固有参数
Ccf=66900;Ccr=62700;Clf=66900;Clr=62700;
% m 为车辆质量,g 为重力加速度,I 为车辆绕 z 轴转动惯量,车辆固有
  参数
m=1723;g=9.8;I=4175;
% 车辆动力学模型输入
dy_dot = -x_dot*phi_dot+2*(Ccf*(delta_f-(y_dot+a*
phi_dot)/x_dot)+Ccr*(b*phi_dot-y_dot)/x_dot)/m;
dx_dot=y_dot*phi_dot+2*(Clf*Sf+Clr*Sr+Ccf*delt
a_f*(delta_f-(y_dot+phi_dot*a)/x_dot))/m;
dphi_dot=(2*a*Ccf*(delta_f-(y_dot+a*phi_dot)/x_
dot)-2*b*Ccr*(b*phi_dot-y_dot)/x_dot)/I;
Y_dot=x_dot*sin(phi)+y_dot*cos(phi);
X_dot=x_dot*cos(phi)-y_dot*sin(phi);
```

```
% 雅克比矩阵求解
f = [dy_dot;dx_dot;phi_dot;dphi_dot;Y_dot;X_dot];
                                        % 动力学模型
kesi = [y_dot,x_dot,phi,phi_dot,Y,X];% 系统状态量
v = delta_f;% 控制量
R = jacobian(f,kesi);% 矩阵 A(t),连续形式
R2 = jacobian(f,v);% 矩阵 B(t),连续形式
% 将转移矩阵由连续形式转换为离散形式,采用一阶差商方法,A = I + T *
  A(t),B = T * B(t)
I = eye(6);% 单位矩阵
syms T;% 采样周期
A = I + T * R;
B = T * R2;
A1 = vpa(A,5);
B1 = vpa(B,5);
```


5.1.2 约束条件建立

在基于动力学的模型预测控制器设计中，除了需要考虑第4章中所加入的控制量约束以及控制增量约束外，还需要添加车辆动力学约束，包括质心侧偏角约束和车辆附着条件约束；同时，由于在动力学模型简化过程中，用到了轮胎的线性近似表达式，因此还需要对轮胎的侧偏角进行限制，以保证行驶过程中轮胎的受力在线性区域中。

（1）质心侧偏角约束

质心侧偏角对车辆的稳定性具有比较大的影响，故必须将质心侧偏角限定在合理的范围内。博世公司（BOSCH）所进行的车辆稳定性研究结果[73,74]显示：在附着良好的干燥沥青路面上，车辆稳定行驶的质心侧偏角极限可以达到±12°；而在附着系数较低的冰雪路面上，极限值近似为±2°。因此，本书将质心侧偏角的约束条件设置为：

$$\begin{aligned} -12^\circ < \beta < 12^\circ (\text{良好路面}) \\ -2^\circ < \beta < 2^\circ (\text{冰雪路面}) \end{aligned} \tag{5.3}$$

（2）车辆附着条件约束

汽车的动力性能不仅仅受到驱动力的制约，还受到轮胎与地面附着条件的限制，因此有必要添加对车辆附着条件的约束。纵向加速度和横向加速度受到

地面附着力的限制，存在如下关系：

$$\sqrt{a_x^2 + a_y^2} \leqslant \mu g \tag{5.4}$$

式中，a_x 和 a_y 分别为纵向加速度和横向加速度。当车辆纵向匀速行驶时，式（5.4）可以进一步简化为：

$$| a_y | \leqslant \mu g \tag{5.5}$$

当行驶路面的附着条件较好时，该项约束条件较为宽松。过大的侧向加速度会影响到人的乘坐舒适性，但约束条件限定过窄会出现运算过程中求解失败的现象。为了综合求解质量和舒适性指标，本书将该约束设定为软约束条件，即求解器会根据每个控制周期的求解情况动态调整该约束条件，如式（5.6）所示：

$$a_{y,\min} - \varepsilon \leqslant a_y \leqslant a_{y,\max} + \varepsilon \tag{5.6}$$

式中，$a_{y,\min}$ 和 $a_{y,\max}$ 为加速度极限约束。

（3）轮胎侧偏角约束

由于所建立的车辆简化动力学模型没有将轮胎侧偏角作为状态量，因此在每一步的控制过程中首先需要对其进行求解。根据式（2.35），质心侧偏角与状态量及控制量存在如下关系：

$$\alpha_f = \frac{\dot{y} + a\dot{\varphi}}{\dot{x}} - \delta_f \qquad \alpha_r = \frac{\dot{y} - b\dot{\varphi}}{\dot{x}}$$

在任意时刻 t，系统各状态量已知，轮胎侧偏角的求解算式为：

$$\alpha_{f,t} = \frac{\dot{y}_t + a\dot{\varphi}_t}{\dot{x}_t} - \delta_{f,t-1} \tag{5.7}$$

$$\alpha_{r,t} = \frac{\dot{y}_t - b\dot{\varphi}_t}{\dot{x}_t} \tag{5.8}$$

根据轮胎的侧偏特性可知，在轮胎侧偏角不超过5°时，侧偏角与侧偏力为线性关系。根据第2章对非线性动力学模型在小角度下的假设，将前轮侧偏角的约束条件做出更加严格的限定：

$$-2.5° < \alpha_{f,t} < 2.5° \tag{5.9}$$

需要将上述约束条件纳入二次规划的求解过程。式（5.7）是关于状态量和控制量的非线性函数，再一次利用线性化方法进行求解。这和之前的系统线性化类似，唯一不同的是约束条件并不需要像输出量那样代入到目标函数中参与最优化计算。以前轮侧偏角为例，线性化采用如下算式：

$$\boldsymbol{C} = \left.\frac{\partial \alpha_f}{\partial \boldsymbol{\xi}}\right|_{\xi(t),u(t-1)}, \boldsymbol{D} = \left.\frac{\partial \alpha_f}{\partial \boldsymbol{u}}\right|_{\xi(t),u(t-1)} \tag{5.10}$$

5.1.3 模型预测控制器设计

在上一章，控制器的目标函数选择了不带松弛因子的形式。这里由于使用的是车辆动力学模型，故复杂程度要远高于运动学模型，同时也加入了更多的约束条件，因此在控制器实际执行过程中，很有可能出现在规定的计算时间内无法得到最优解的情况。因此，有必要在目标函数中加入松弛因子，表达式如下：

$$J(\boldsymbol{\xi}_{\mathrm{dyn}}(t),\boldsymbol{u}_{\mathrm{dyn}}(t-1),\Delta\boldsymbol{U}_{\mathrm{dyn}}(t)) =$$

$$\sum_{i=1}^{N_p} \| \boldsymbol{\eta}_{\mathrm{dyn}}(t+i\mid t) - \boldsymbol{\eta}_{\mathrm{dyn,ref}}(t+i\mid t) \|_{\boldsymbol{Q}}^2 + \sum_{i=1}^{N_c-1} \| \Delta\boldsymbol{u}_{\mathrm{dyn}}(t+i\mid t) \|_{\boldsymbol{R}}^2 + \rho\varepsilon^2$$

综合上面的目标函数和约束条件，基于动力学模型的主动转向跟踪控制器在每个控制周期内要解决如下的优化问题：

$$\begin{aligned}
&\min_{\Delta\boldsymbol{U}_{\mathrm{dyn}},\varepsilon} \sum_{i=1}^{N_p} \| \boldsymbol{\eta}_{\mathrm{dyn}}(t+i\mid t) - \boldsymbol{\eta}_{\mathrm{dyn,ref}}(t+i\mid t) \|_{\boldsymbol{Q}}^2 + \\
&\qquad \sum_{i=1}^{N_c-1} \| \Delta\boldsymbol{u}_{\mathrm{dyn}}(t+i\mid t) \|_{\boldsymbol{R}}^2 + \rho\varepsilon^2 \\
&\text{s.t. } \Delta\boldsymbol{U}_{\mathrm{dyn,min}} \leqslant \Delta\boldsymbol{U}_{\mathrm{dyn},t} \leqslant \Delta\boldsymbol{U}_{\mathrm{dyn,max}} \\
&\qquad \boldsymbol{U}_{\mathrm{dyn,min}} \leqslant \boldsymbol{A}\Delta\boldsymbol{U}_{\mathrm{dyn},t} + \boldsymbol{U}_{\mathrm{dyn},t} \leqslant \boldsymbol{U}_{\mathrm{dyn,max}} \\
&\qquad \boldsymbol{y}_{\mathrm{hc,min}} \leqslant \boldsymbol{y}_{\mathrm{hc}} \leqslant \boldsymbol{y}_{\mathrm{hc,max}} \\
&\qquad \boldsymbol{y}_{\mathrm{sc,min}} - \varepsilon \leqslant \boldsymbol{y}_{\mathrm{sc}} \leqslant \boldsymbol{y}_{\mathrm{hc,max}} + \varepsilon \\
&\qquad \varepsilon > 0
\end{aligned} \tag{5.11}$$

式中，$\boldsymbol{y}_{\mathrm{hc}}$为硬约束输出，即不能放宽约束范围的输出量；$\boldsymbol{y}_{\mathrm{sc}}$为软约束输出，即可以通过松弛因子进行动态调整约束范围的输出量；$\boldsymbol{y}_{\mathrm{hc,min}}$和$\boldsymbol{y}_{\mathrm{hc,max}}$为硬约束极限值；$\boldsymbol{y}_{\mathrm{sc,min}}$和$\boldsymbol{y}_{\mathrm{sc,max}}$为软约束极限值。

在每个控制周期内完成式（5.11）的求解后，得到了控制时域内的一系列控制输入增量和松弛因子：

$$\Delta\boldsymbol{U}_{\mathrm{dyn},t}^{*} = [\Delta\boldsymbol{u}_{\mathrm{dyn},t}^{*},\Delta\boldsymbol{u}_{\mathrm{dyn},t+1}^{*},\cdots,\Delta\boldsymbol{u}_{\mathrm{dyn},t+N_c-1}^{*},\varepsilon]^{\mathrm{T}} \tag{5.12}$$

将该控制序列中第一个元素作为实际的控制输入增量作用于系统，即：

$$\boldsymbol{u}_{\mathrm{dyn}}(t) = \boldsymbol{u}_{\mathrm{dyn}}(t-1) + \Delta\boldsymbol{u}_{\mathrm{dyn},t}^{*} \tag{5.13}$$

进入下一个控制周期后，重复上述过程，如此循环实现对期望轨迹的跟踪控制。

5.2 联合仿真平台搭建

有了第4章的基础，完成基于动力学模型的主动转向控制器的设计将会更加容易。整个联合仿真平台依然保持着原来的形式（如图5.1所示），主要由CarSim提供的车辆动力学模块和S函数编写的控制器模块组成。而在CarSim中主要变化在于车辆模型的建立和输入、输出接口设定，在控制器模块中需要根据新的动力学模型进行调整。以下就分别从这两方面进行介绍。

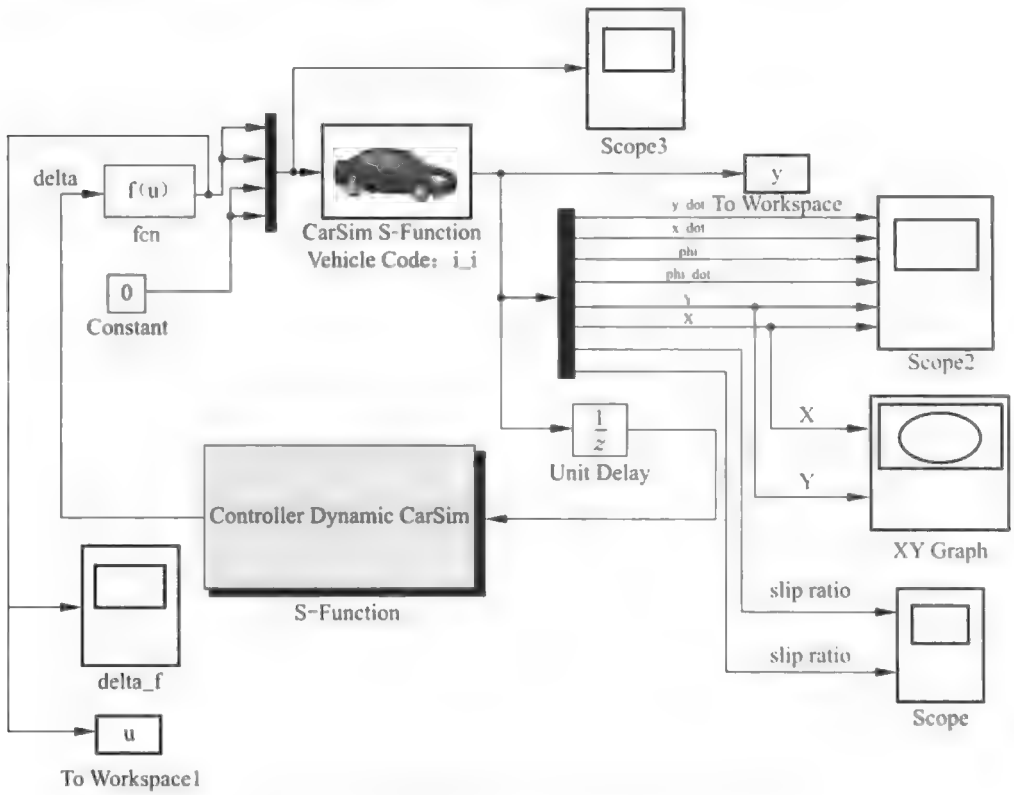


图5.1 基于动力学模型的CarSim/Simulink联合仿真平台

5.2.1 在CarSim中建立车辆模型

本章中需要使用的车辆动力学参数如表5.1所示，主要为车辆尺寸信息和轮胎信息。因此，我们在设置参数时首先进入车辆尺寸参数界面，其设置结果如图5.2所示。车辆的轮胎选择225/60 R18，前后悬架都选择为独立悬架，其

设置结果如图5.3所示：

表5.1　车辆动力学参数

参数	取　值
车辆质量/kg	$m=1\,723$
绕 z 轴转动惯量/(kg·m^2)	$I_z=4\,175$
车身尺寸/m	$L=4$，$W=1.988$
轴距/m	$a=1.232$，$b=1.468$
轮胎侧偏刚度/(N·rad^{-1})	$C_{cf}=66\,900$，$C_{cr}=62\,700$

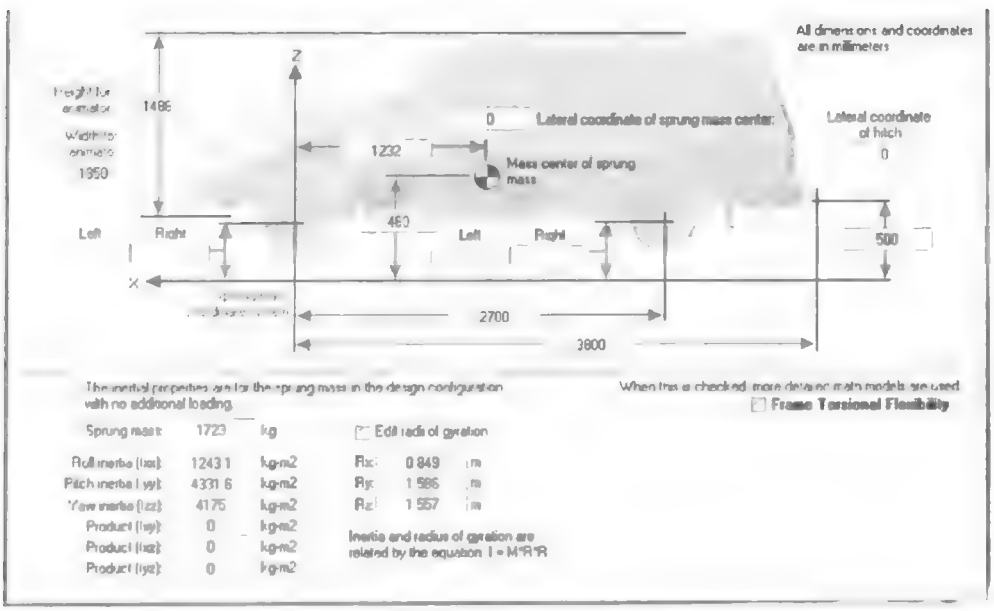


图5.2　车辆尺寸参数设置

在输入输出接口方面，由于只对车辆的前轮进行控制，因此CarSim的输入端口选择车辆4个轮胎的转角，其中2个前轮的转角由控制器提供，2个后轮转角固定为0，其设置如图5.4所示。输出端口包括动力学系统的6个状态量和2个前轮的质移率，共8个输出量，其设置如图5.5所示。

5.2.2　控制程序编写

在第4章中，我们已经采用S函数编写了基于车辆运动学模型的轨迹跟踪控制器，在这一章中，需要继续用S函数建立控制器，完成的程序为chapter5_2_2.m。在主函数中，仍然和第4章保持一致，通过标志位调用不同的函

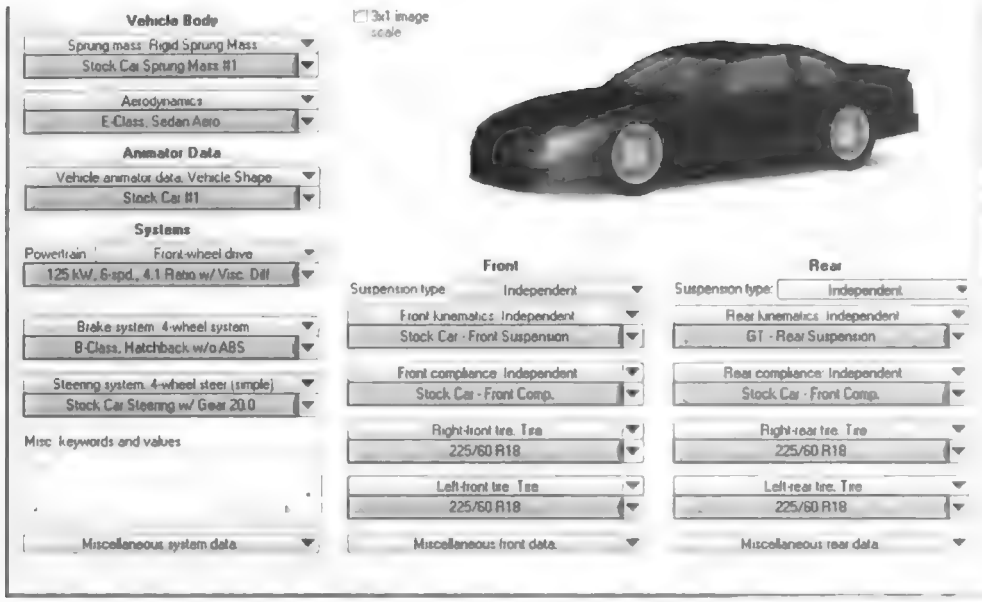


图 5.3　车辆悬架及轮胎设置

Active Import Variables

	Name	Mode	Initial Value
1	IMP_STEER_L1	Add	0.0
2	IMP_STEER_R1	Add	0.0
3	IMP_STEER_L2	Add	0.0
4	IMP_STEER_R2	Add	0.0

图 5.4　CarSim 输入端口设置

Active Variables for Export

1. VyBf_SM
2. VxBf_SM
3. Yaw
4. AV_Y
5. Yo
6. Xo
7. Kappa_L1
8. Kappa_R1

图 5.5　CarSim 输出端口设置

数。在初始化函数中，主要改变了输入与输出参数的个数，相应地需要对系统状态量进行初始化。

```
function [sys,x0,str,ts]=mdlInitializeSizes

% Call simsizes for a sizes structure,fill it in,and con-
  vert it
% to a sizes array.

sizes=simsizes;
sizes.NumContStates  =0;
sizes.NumDiscStates  =6;
sizes.NumOutputs     =1;
sizes.NumInputs      =8;
sizes.DirFeedthrough=1;% Matrix D is non-empty.
sizes.NumSampleTimes=1;
sys=simsizes(sizes);
x0=[0.001;0.0001;0.0001;0.00001;0.00001;0.00001];
global U;
U=[0];% 控制量初始化
% global x;
% x=zeros(md.ne+md.pye+md.me+md.Hu*md.me,1);
% Initialize the discrete states.
str=[];              % Set str to an empty matrix.
ts  =[0.050];        % sample time:[period,offset]
% End of mdlInitializeSizes
```

程序的主体为函数 mdlOutputs（），需要完成程序基本参数初始化、车辆参数输入、参考轨迹生成、二次规划问题求解等几个模块。

首先介绍第一部分，也即程序参数初始化，主要功能是设置模型预测控制器的基本参数，包括输入与输出个数、预测和控制时域等。由于 CarSim 中采用的速度单位为“km/h”，角度单位为“°”，而我们在 S 函数中采用的单位分别为“m/s”和“rad”，因此需要加入对输入参数的转换，而这一点往往也是我们最容易忽视的。

```
global a b;
global U;
tic
Nx=6;% 状态量的个数
Nu=1;% 控制量的个数
Ny=2;% 输出量的个数
Np=20;% 预测步长
Nc=5;% 控制步长
T=0.02;% 仿真步长
% 输入接口转换,x_dot 后面加一个非常小的数,是防止出现分母为零的
  情况
y_dot=u(1)/3.6;
x_dot = u(2)/3.6 + 0.0001; % CarSim 输出的是 km/h, 转换
  为 m/s
phi=u(3)*3.141592654/180;% CarSim 输出的为角度,角度转换为
  弧度
phi_dot=u(4)*3.141592654/180;
Y=u(5);% 单位为 m
X=u(6);% 单位为 m
Y_dot=u(7);
X_dot=u(8);
```


预测模型是模型预测控制的3项基本原理之一。在控制器中需要根据我们采用的动力学模型设置参数，包括车辆轴距、轮胎刚度以及车辆质量等。具体代码如下：

```
%% 车辆参数输入
% 前后车轮的滑移率
Sf=0.2;Sr=0.2;
% 前后车轮距离车辆质心的距离,车辆固有参数
lf=1.232;lr=1.468;
```

```
% 分别为前后车轮的纵横向侧偏刚度，车辆固有参数
Ccf=66900;Ccr=62700;Clf=66900;Clr=62700;
% m 为车辆质量，g 为重力加速度，I 为车辆绕 Z 轴的转动惯量，车辆固有
  参数
m=1723;g=9.8;I=4175;
```

完成车辆动力学参数设置后，需要给出参考轨迹。在仿真过程中，既可以将参考轨迹生成的部分放在控制器中，也可以单独提供一个模块并建立与控制器的接口。本文将参考轨迹生成的部分放在控制器中，而有兴趣的读者也可以自己尝试外加模块，以实现同样功能。参考轨迹的含义将在仿真结果部分介绍，以下仅给出实现代码：

```
% 参考轨迹生成
shape=2.4;  % 参数名称，用于参考轨迹生成
dx1=25;dx2=21.95;
dy1=4.05;dy2=5.7;
Xs1=27.19;Xs2=56.46;% 参数名称
X_predict=zeros(Np,1);% 用于保存预测时域内的纵向位置信息，这是
                        计算期望轨迹的基础
y_ref=zeros(Np,1);% 用于保存预测时域内的期望轨迹
Y_ref=zeros(Np,1);% 用于保存预测时域内的期望轨迹
dphi_ref=zeros(Np,1);
```

上述步骤都可以被看成建立模型预测控制器的准备工作，而线性时变模型预测控制的主要工作是根据线性时变模型预测系统输出，结合系统约束，将车辆轨迹跟踪问题转换为标准二次规划问题进行求解。线性时变模型预测控制的算式推导部分可以参考本书3.3节。读者在完成了3.3.3中的实例程序后，再结合本章中对动力学模型的线性化求解，就可以比较顺利地完成以下程序：

```
% 矩阵转换，即状态量与控制量合在一起
kesi=zeros(Nx+Nu,1);
kesi(1)=y_dot;% u(1)==X(1)
kesi(2)=x_dot;% u(2)==X(2)
```

```
kesi(3)=phi;% u(3)==X(3)
kesi(4)=phi_dot;
kesi(5)=Y;
kesi(6)=X;
kesi(7)=U(1);
delta_f=U(1);
% ================================================================
% 权重矩阵设置
% ================================================================
Q_cell=cell(Np,Np);
for i=1:1:Np;
      for j=1:1:Np;
            if i==j
                Q_cell{i,j}=[2000   0;0 10000;];
            else
                Q_cell{i,j}=zeros(Ny,Ny);
            end
      end
end
R=5*10^5*eye(Nu*Nc);
Row=1000;% 松弛因子权重
% ================================================================
% 系统转移矩阵设置
% ================================================================
a=[1-(259200*T)/(1723*x_dot),-T*(phi_dot+(2*
   ((460218*phi_dot)/5-62700*y_dot))/(1723*x_dot^2)-
   (133800*((154*phi_dot)/125+y_dot))/(1723*x_dot^2)),
   0,-T*(x_dot-96228/(8615*x_dot)),0,0

T*(phi_dot-(133800*delta_f)/(1723*x_dot)),(133800*T*
    delta_f*((154*phi_dot)/125+y_dot))/(1723*x_dot^2)
    +1,0,T*(y_dot-(824208*delta_f)/(8615*x_dot)),0,0

0,0,1,T,0,0
```

```
(33063689036759 * T)/(7172595384320 * x_dot), T *
(((2321344006605451863 * phi_dot)/8589934592000 -
(2[illegible]8028897689 * y_dot)/34359738368)/(4175 * x_dot^2) +
(5663914248162509 * ((154 * phi_dot)/125 + y_dot))/(1
43451907686400 * x_dot^2)),0,1, - (813165919007900927 *
T)/(7172595384320000 * x_dot),0,0

T * cos(phi),T * sin(phi), T * (x_dot * cos(phi) - y_dot * sin
(phi)),0,1,0

- T * sin(phi),T * cos(phi), - T * (y_dot * cos(phi) + x_dot *
sin(phi)),0,0,1];

b = [                                          133800 * T/1723
T * ((267600 * delta_f)/1723 - (133800 * ((154 * phi_dot)/
125 + y_dot))/(1723 * x_dot))
                                                          0
                    5663914248162509 * T/143451907686400
                                                          0
                                                          0];

A_cell = cell(2,2);
B_cell = cell(2,1);
A_cell{1,1} = a;
A_cell{1,2} = b;
A_cell{2,1} = zeros(Nu,Nx);
A_cell{2,2} = eye(Nu);
B_cell{1,1} = b;
B_cell{2,1} = eye(Nu);
A = cell2mat(A_cell);
B = cell2mat(B_cell);
C = [0 0 1 0 0 0 0;0 0 0 0 1 0 0;];
%
% 计算偏差
%
```

```
d_k = zeros(Nx,1);
state_k1 = zeros(Nx,1);% 预测的下一时刻状态量,用于计算偏差
% 根据离散非线性模型预测下一时刻状态量
state_k1(1,1) = y_dot + T * ( - x_dot * phi_dot + 2 * (Ccf *
      (delta_f - (y_dot + lf * phi_dot)/x_dot) + Ccr * (lr * phi
      _dot - y_dot)/x_dot)/m);
state_k1(2,1) = x_dot + T * (y_dot * phi_dot + 2 * (Clf * Sf + Clr
      * Sr + Ccf * delta_f * (delta_f - (y_dot + phi_dot * l
      f)/x_do t))/m);
state_k1(3,1) = phi + T * phi_dot;
state_k1(4,1) = phi_dot + T * ((2 * lf * Ccf * (delta_f - (y_dot
      + lf * phi_dot)/x_dot) - 2 * lr * Ccr * (lr * phi_dot - y_d
      ot)/x_d ot)/I);
state_k1(5,1) = Y + T * (x_dot * sin(phi) + y_dot * cos(phi));
state_k1(6,1) = X + T * (x_dot * cos(phi) - y_dot * sin(phi));
d_k = state_k1 - a * kesi(1:6,1) - b * kesi(7,1);
d_piao_k = zeros(Nx + Nu,1);% d_k 的增广形式
d_piao_k(1:6,1) = d_k;
d_piao_k(7,1) = 0;

PSI_cell = cell(Np,1);
THETA_cell = cell(Np,Nc);
GAMMA_cell = cell(Np,Np);
PHI_cell = cell(Np,1);
for p = 1:1:Np;
     PHI_cell{p,1} = d_piao_k;
     for q = 1:1:Np;
          if q <= p;
              GAMMA_cell{p,q} = C * A^(p - q);
          else
              GAMMA_cell{p,q} = zeros(Ny,Nx + Nu);
          end
       end
   end
```

```
    for j = 1:1:Np
    PSI_cell{j,1} = C * A^j;
        for k = 1:1:Nc
            if k <= j
                THETA_cell{j,k} = C * A^(j - k) * B;
            else
                THETA_cell{j,k} = zeros(Ny,Nu);
            end
        end
end
PSI = cell2mat(PSI_cell);% size(PSI) = [Ny * Np Nx + Nu]
THETA = cell2mat(THETA_cell);% size(THETA) = [Ny * Np Nu *
Nc]
GAMMA = cell2mat(GAMMA_cell);% 大写的 GAMMA
PHI = cell2mat(PHI_cell);
Q = cell2mat(Q_cell);
H_cell = cell(2,2);
H_cell{1,1} = THETA' * Q * THETA + R;
H_cell{1,2} = zeros(Nu * Nc,1);
H_cell{2,1} = zeros(1,Nu * Nc);
H_cell{2,2} = Row;
H = cell2mat(H_cell);
error_1 = zeros(Ny * Np,1);
Yita_ref_cell = cell(Np,1);
%
% 获得局部参考轨迹,注意防止越界问题
%
T_all = 20;% 总的仿真时间,主要功能是防止计算期望轨迹越界
for p = 1:1:Np
    if t + p * T > T_all
        X_DOT = x_dot * cos(phi) - y_dot * sin(phi);% 惯性坐
标系下纵向速度
        X_predict(Np,1) = X + X_DOT * Np * T;
        z1 = shape/dx1 *(X_predict(Np,1) - Xs1) - shape/2;
```

```
        z2 = shape/dx2 *(X_predict(Np,1)-Xs2)-shape/2;
        Y_ref(p,1)= dy1/2 *(1 + tanh(z1))- dy2/2 *(1 + tanh
          (z2));
        phi_ref(p,1)= atan(dy1 * (1/cosh(z1))^2 * (1.2/dx1)-
            dy2 * (1/cosh(z2))^2 * (1.2/dx2));
        Yita_ref_cell{p,1} =[phi_ref(p,1);Y_ref(p,1)];
    else
        X_DOT = x_dot * cos(phi)- y_dot * sin(phi);% 惯性坐标系
                                                    下纵向速度
        X_predict(p,1)= X+ X_DOT * p * T;% 首先计算出未来 X 的位
                                          置,X(t)= X + X_dot * t
        z1 = shape/dx1 * (X_predict(p,1) - Xs1) - shape/2;
        z2 = shape/dx2 * (X_predict(p,1) - Xs2) - shape/2;
        Y_ref(p,1)= dy1/2 * (1 + tanh(z1)) - dy2/2 * (1 + tanh
          (z2));
        phi_ref(p,1) = atan(dy1 * (1/cosh(z1))^2 * (1.2/dx1)
            - dy2 * (1/cosh(z2))^2 * (1.2/dx2));
        Yita_ref_cell{p,1} = [phi_ref(p,1);Y_ref(p,1)];
    end
end
Yita_ref = cell2mat(Yita_ref_cell);
error_1 = Yita_ref - PSI * kesi - GAMMA * PHI;% 求偏差
f_cell = cell(1,2);
f_cell{1,1}=2 * error_1' * Q * THETA;
f_cell{1,2}=0;
f = -cell2mat(f_cell);
```

在上面的程序中，我们已经获得了二次规划问题中至关重要的两个矩阵 H 和 f，而接下来就是要根据需要的约束设定好约束条件，包括控制量约束、控制增量约束和输出量约束。

```
%% 以下为约束生成区域
% ================================================
```

```
% 控制量约束
% ================================================================
A_t = zeros(Nc,Nc);
    for p = 1:1:Nc
        for q = 1:1:Nc
            if q <= p
                A_t(p,q)=1;
            else
                A_t(p,q) = 0;
            end
        end
    end
A_I = kron(A_t,eye(Nu));% 求克罗内克积
Ut = kron(ones(Nc,1),U(1));
umin = -0.1744;% 维数与控制变量的个数相同
umax = 0.1744;
delta_umin = -0.0148;
delta_umax = 0.0148;
Umin = kron(ones(Nc,1),umin);
Umax = kron(ones(Nc,1),umax);
% ================================================================
% 输出量约束
% ================================================================
ycmax = [0.21;5];
ycmin = [-0.3;-3];
Ycmax = kron(ones(Np,1),ycmax);
Ycmin = kron(ones(Np,1),ycmin);
% ================================================================
% 结合控制量约束和输出量约束
% ================================================================
A_cons_cell = {A_I zeros(Nu * Nc,1); - A_I zeros(Nu * Nc,1);
 THETA zeros(Ny * Np,1); - THETA zeros(Ny * Np,1)};
b_cons_cell = {Umax - Ut; - Umin + Ut;Ycmax - PSI * kesi - GA
 MMA * PHI; - Ycmin + PSI * kesi + GAMMA * PHI};
```

```
A_cons = cell2mat(A_cons_cell);% (求解方程)状态量不等式约束增益矩阵
b_cons = cell2mat(b_cons_cell);% (求解方程)状态量不等式约束的取值
% 状态量约束
M = 10;
delta_Umin = kron(ones(Nc,1),delta_umin);
delta_Umax = kron(ones(Nc,1),delta_umax);
lb = [delta_Umin;0];% (求解方程)状态量下界,包含控制时域内控制增量和松弛因子
ub = [delta_Umax;M];% (求解方程)状态量上界,包含控制时域内控制增量和松弛因子
```

最后，结合二次规划的标准矩阵和约束条件进行最优求解，需要完成的工作包括选择优化方法（一般在有效集与内点法之间选择）、设定初始点等。求解结束后，根据模型预测控制的基本原理，将控制增量序列的第一个元素与上一个时刻的控制量相加，输出给 CarSim 动力学模块。


```
options = optimset('Algorithm','active-set');% 选取有效集法
x_start = zeros(Nc+1,1);
[X,fval,exitflag] = quadprog(H,f,A_cons,b_cons,[],[],lb,ub,x_start,options);
% % 计算输出
u_piao = X(1);% 得到控制增量
U(1) = kesi(7,1) + u_piao;% 当前时刻的控制量为上一时刻控制+控制增量
sys = U;% 系统输出
% 程序结束
```

至此，我们就完成了基于动力学模型的主动转向控制器的设计。读者可以分段将程序读懂，然后将各段程序合在一起，在联合仿真平台上测试程序的功能。以下将介绍本书针对无人驾驶车辆的行驶工况所设计的仿真环境及验证结果。

5.3 仿真验证

5.3.1 参考轨迹选择

基于动力学模型的模型预测控制器主要用于较高速度下的轨迹跟踪。在这种情况下对于控制器的评价不仅仅是跟踪精度，还需要重点考虑跟踪过程中的稳定性。在普通车辆行驶稳定性的测试中，双移线工况是使用频率较高的一种测试路段[76,77]。国内外也有较多的学者以双移线轨迹进行无人驾驶车辆轨迹跟踪能力的测试[78-80]。因此，本文参考文献[77]中的双移线轨迹，对所设计的模型预测控制器进行仿真测试。

双移线轨迹如图5.6所示，由参考横向位置Y_{ref}和参考横摆角φ_{ref}构成。Y_{ref}和φ_{ref}都表示为关于横向位置X的非线性函数，具体算式如下：

$$Y_{ref}(X)=\frac{d_{y1}}{2}(1+\tanh(z_1))-\frac{d_{y2}}{2}(1+\tan(z_2)) \tag{5.14}$$

$$\varphi_{ref}(X)=\arctan\left(d_{y1}\left(\frac{1}{\cosh(z_1)}\right)^2\left(\frac{1.2}{d_{x1}}\right)-d_{y2}\left(\frac{1}{\cosh(z_2)}\right)^2\left(\frac{1.2}{d_{x2}}\right)\right) \tag{5.15}$$

式中，$z_1=\frac{2.4}{25}(X-27.19)-1.2$，$z_2=\frac{2.4}{21.95}(X-56.46)-1.2$，$d_{x1}=25$，$d_{x2}=21.95$，$d_{y1}=4.05$，$d_{y2}=5.7$。

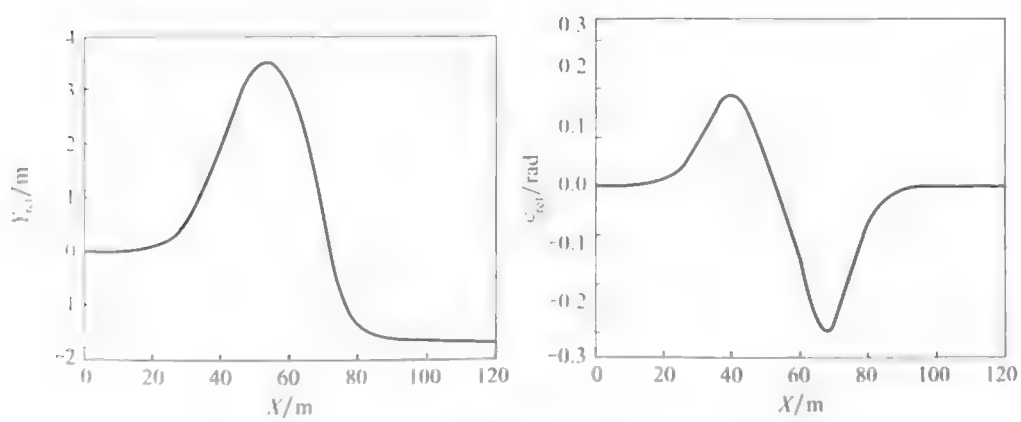


图5.6 双移线参考轨迹

5.3.2 不同仿真工况下的仿真结果

所设计的主动转向跟踪控制器以车辆前轮偏角作为控制器的输入，控制目标是通过不断减少与参考轨迹的偏差，跟踪参考轨迹。为了更加全面地分析所提出控制方法的可行性，本书设计3种不同的仿真工况，尽可能体现控制器在实际运用时可能遇到的各种情况。

（1）仿真工况1：控制系统对路面附着条件的鲁棒性

无人驾驶车辆在不同附着条件的道路上（如干燥路面、湿滑路面）行驶时，车辆自身的动力学参数，如轮胎侧偏刚度等将会发生变化，同时也会出现地面提供的侧向力不足的情况。这给控制器的性能带来了一定的挑战。仿真工况1就是在不同附着条件的道路上对车辆轨迹跟踪能力进行测试。无人驾驶车辆正常行驶路面的附着系数为$\mu=0.7\sim1$，湿滑路面的附着系数为$\mu=0.4$左右。因此，分别选取$\mu=0.4$，0.8两种情况进行测试。

车辆纵向速度设定为30 m/s，轨迹跟踪控制器的基本参数设定为：$T=0.05$ s，$N_p=25$，$N_c=15$，$-10°\leqslant\delta_f\leqslant10°$，$-0.85°\leqslant\Delta\delta_f\leqslant0.85°$，权重矩阵设置为：

$$\boldsymbol{Q}=\begin{bmatrix}200&0&0\\0&100&0\\0&0&100\end{bmatrix},\quad \boldsymbol{R}=5\times10^{4},\quad \rho=1\,000$$

仿真结果如图5.7所示。其中，图5.7（a）为系统输出量$[Y,\varphi]$随时间的变化关系，图5.7（b）为系统控制量δ_f和控制增量$\Delta\delta_f$随时间的变化关系，图5.7（c）为所设定的动力学约束条件在仿真过程中的变化。从图5.7（a）中曲线可以看出，控制器在不同附着条件的路面上都能很好地跟踪期望轨迹，但附着条件良好时车辆的跟踪误差能够进一步缩小。若附着条件较差，地面不能提供足够的侧向力，车辆转向时横摆角会出现较大偏差（图5.7（a）中70 m处），轨迹跟踪控制器能够及时修正偏差，最终将偏差收敛为0。从图5.7（b）中可以看出，跟踪过程中车辆的控制量以及控制增量都处于约束范围内，保证了给定车辆的控制量能被执行机构顺利执行。图5.7（c）中，除了设定为软约束的纵向加速度外，各项约束都被限定在了给定的区间范围内。软约束的加入保证系统能够在给定时间内得到可行解。综合上述分析，控制系统能够在不同附着条件下，较好地跟踪期望轨迹，并具备良好的稳定性。

（2）仿真工况2：控制算法对速度的鲁棒性

很多控制算法往往需要针对不同的行驶速度确定不同的控制参数，而模型预测控制器能够根据建立的车辆模型预测系统未来的输出，对车速的变化具有

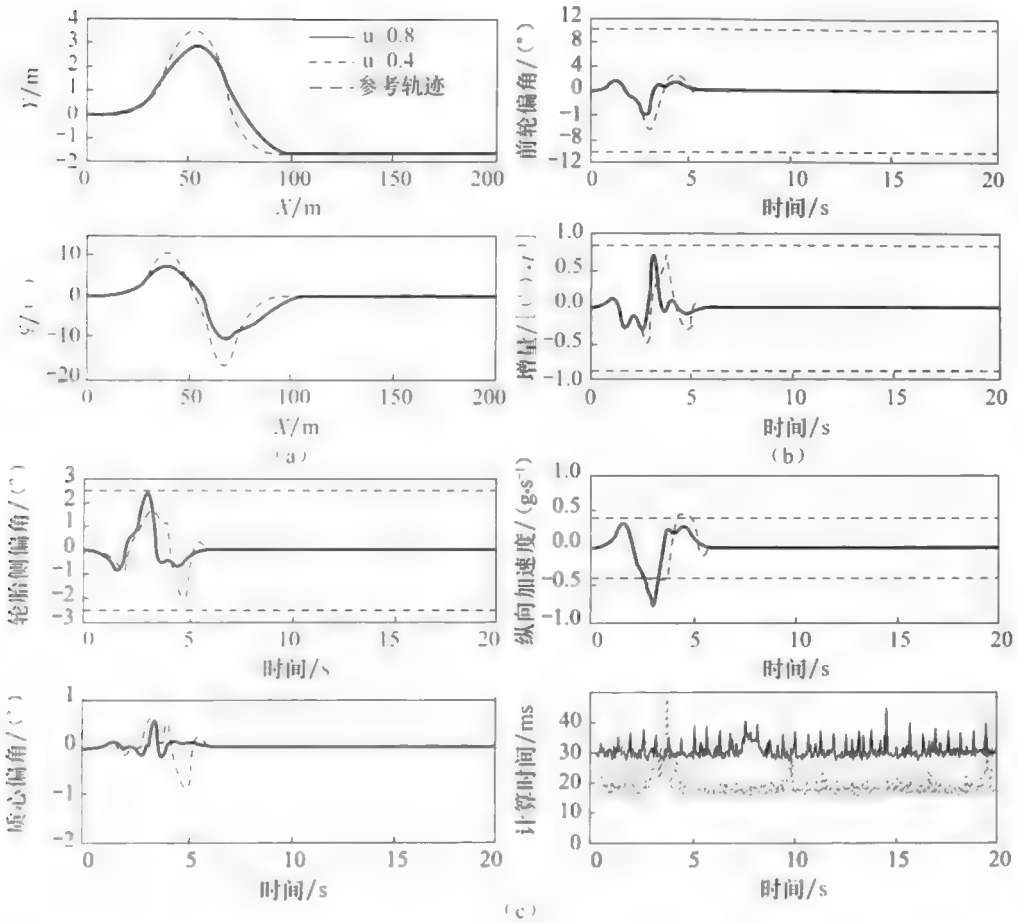


图 5.7 仿真工况 1 的仿真结果

(a) 输出量随时间变化历程；(b) 控制量随时间变化历程；(c) 约束量随时间变化历程

很强的鲁棒性。仿真工况 2 就是在不同的车速下，以相同参数的控制器实现对无人驾驶车辆的主动转向控制，分析控制器对不同车速的鲁棒性。

主动转向仿真实验分别在 10 m/s、20 m/s 和 30 m/s 的速度下进行，道路附着条件良好，$\mu = 0.8$。控制器所采用的参数为：$T = 0.05$ s，$N_p = 25$，$N_c = 10$，$-10° \leqslant \delta_f \leqslant 10°$，$-0.85° \leqslant \Delta\delta_f \leqslant 0.85°$，权重矩阵设置为：

$$\boldsymbol{Q} = \begin{bmatrix} 200 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{bmatrix}, \quad \boldsymbol{R} = 1.1 \times 10^5, \quad \rho = 1\,000$$

仿真工况 2 的仿真结果如图 5.8 所示。其中，图 5.8 (a) 为 3 种不同车速下车辆的轨迹跟踪对比，图 5.8 (b) 为控制量和控制增量随时间变化曲线，

图 5.8（c）为动力学约束随时间变化曲线，实线、点线和点画线分别对应车速为 10 m/s、20 m/s 和 30 m/s 的仿真结果。从图 5.8（a）可以看出，在相同控制参数下，不同速度行驶的车辆都具备良好的轨迹跟踪性能，体现出对速度很强的鲁棒性。图 5.8（b）中，不同速度下控制量的增量差异较大，速度越高，控制量增量就越大，但始终保持在约束范围内。图 5.8（c）中各动力学约束保持在给定的区间范围内，质心侧偏角则远低于极限范围，表明车辆行驶过程非常平稳。因此，本书所设计的轨迹跟踪控制器能够在不同车速下完成对期望轨迹的跟踪，车速增加并不会导致车辆稳定性能的下降。

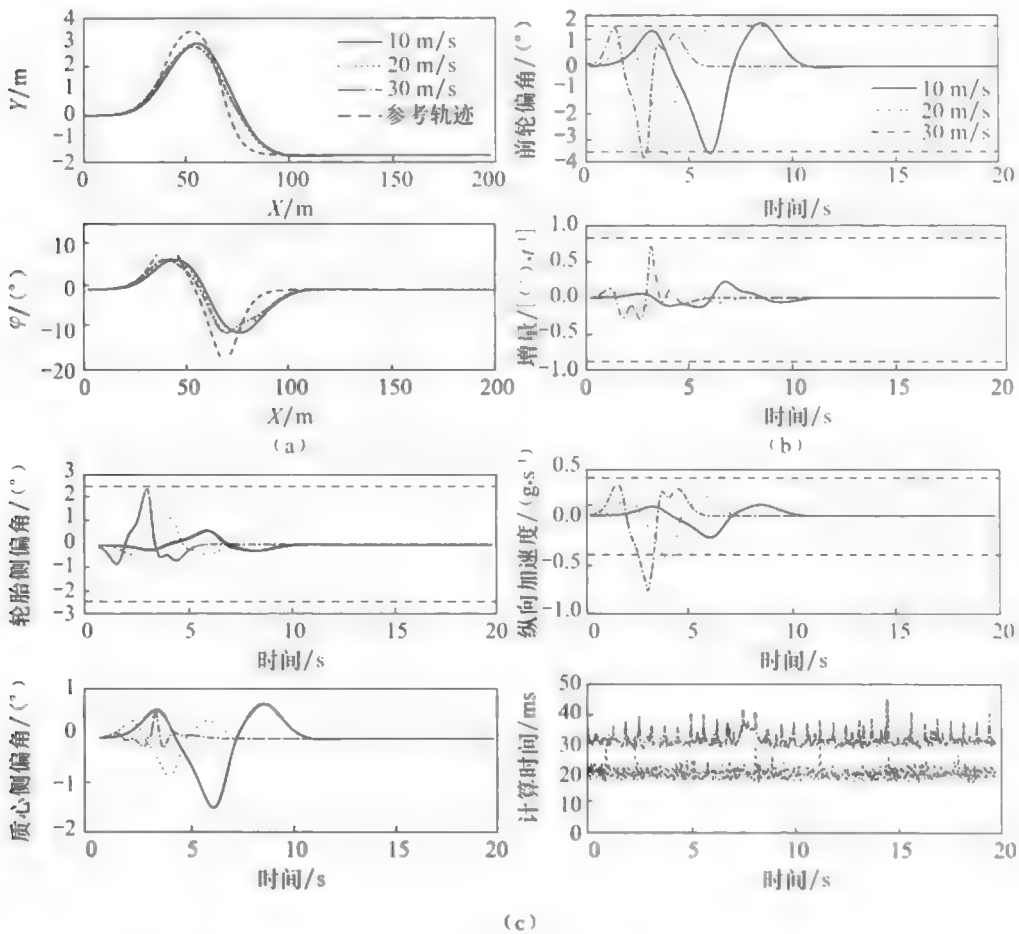


图 5.8　仿真工况 2 的仿真结果

（a）输出量随时间变化历程；（b）控制量随时间变化历程；（c）约束量随时间变化历程

（3）仿真工况 3：不同设计参数对控制器的影响

不同参数对控制器性能的影响有所差异。对于模型控制算法来说，最重要

的设计参数就是预测时域与控制时域。为了考察不同时域对控制器跟踪性能和计算量的影响，分别设计了控制器 A 和控制器 B。控制器 A 所采用的参数如下：$T=0.05$ s，$N_p=25$，$N_c=10$，$-10°\leqslant\delta_f\leqslant10°$，$-0.85°\leqslant\Delta\delta_f\leqslant0.85°$，$\mu=0.8$，权重矩阵设置为：

$$\boldsymbol{Q}=\begin{bmatrix}200 & 0 & 0\\ 0 & 100 & 0\\ 0 & 0 & 100\end{bmatrix},\quad \boldsymbol{R}=5\times10^4,\quad \rho=1\,000$$

相比控制器 A，控制器 B 减少了预测时域和控制时域的步长，设定 $N_p=15$，$N_c=1$，其余参数与控制器 A 保持一致。分别在不同的控制器作用下，验证无人驾驶车辆轨迹跟踪及保持稳定行驶的能力。

仿真工况 3 的仿真结果如图 5.9 所示。其中，图 5.9（a）为两个控制跟踪期望轨迹的对比，图 5.9（b）为两个控制器在每个周期内的计算时间，图中实线和点线分别对应控制器 A、B 的仿真结果。从图 5.9（a）可以看出，控制器 B 能够跟踪期望轨迹，并且最终能够将偏差收敛为 0。与控制器 A 的跟踪效果相比，控制器 B 的跟踪偏差较大，尤其是对期望横摆角的跟踪；但控制器 B 并不是一无是处，图 5.9（b）中的曲线可以非常直观地表示出，控制器 B 的计算时间要远低于控制器 A，平均只有不到 10ms。因此，采用较小的预测时域和控制时域对于提升控制系统的实时性效果明显。在硬件条件较低而又期望保证系统的实时性情况下，可以采用这样的设计方法。

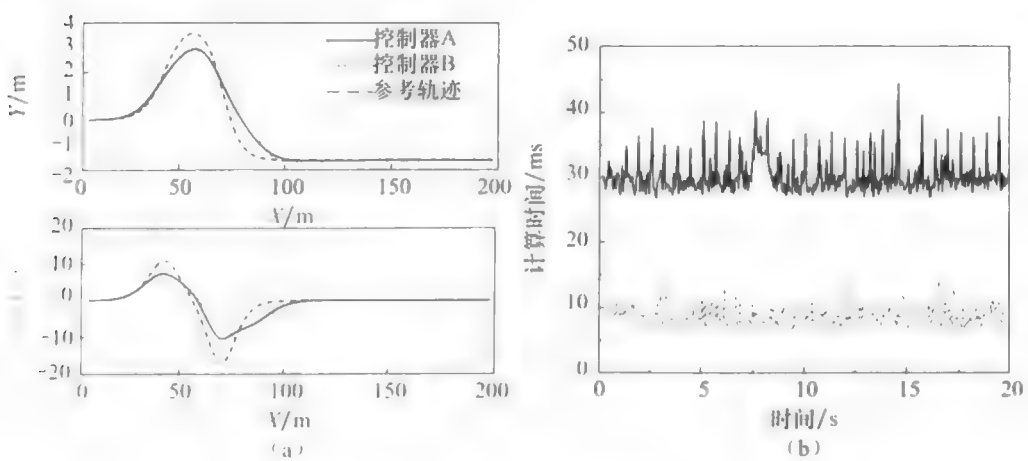


图 5.9 仿真工况 3 的仿真结果

（a）输出量随时间变化关系；（b）不同控制器的计算时间

第 6 章

加入规划层的轨迹跟踪控制

本章介绍地面无人驾驶车辆结合路径规划层的轨迹跟踪控制器。车辆类型选择前轮转向的乘用汽车。首先介绍了基于车辆点质量模型的 MPC 轨迹规划控制器，规划出满足车辆动力学约束并实现避障的可行轨迹；然后介绍了在车辆动力学模型的基础上的 MPC 路径跟踪控制器，并给出了 S 函数代码；最后详细介绍了利用 Simulink/CarSim 进行联合仿真的实例。

无人驾驶车辆在实际环境中进行自主驾驶时，由于外部环境是动态的、变化的，所以给定期望轨迹下的跟踪控制并不能保证无人驾驶车辆准确地处理任何轨迹跟踪问题。原因如下：

① 由于环境未知，预先给定的参考路径不一定能满足安全行驶的要求。

② 当所给定的期望轨迹存在障碍物时，无人驾驶车辆要根据障碍物信息进行重新规划，绕开障碍物后再继续跟踪期望轨迹。

③ 为了减少计算量，在路径跟踪过程中使用的车辆模型是经过简化的，给定的参考轨迹不一定能满足车辆行驶的动力学和运动学的约束条件。

因此本章从轨迹重规划的角度出发，在轨迹跟踪层之上建立轨迹重规划层，也即局部规划层。该规划层能够根据传感器获得的障碍物信息和参考路径信息重新规划出绕开障碍物的局部期望轨迹信息，再将局部期望轨迹信息输入跟踪控制层，在实现避让障碍物的同时，实现对全局参考路径的跟踪。

6.1 结合规划层的轨迹跟踪控制系统

在原有的轨迹跟踪控制基础上加入规划层后，形成新的控制系统，结构

如图 6.1 所示，该控制系统主要由带避障功能的轨迹重规划模块和跟踪控制模块构成。跟踪控制模块接收来自规划层的局部参考轨迹，输出前轮偏角控制量；而轨迹重规划模块接收来自传感器的障碍物信息以及来自全局规划的参考轨迹信息，通过模型预测控制算法规划出局部参考轨迹，再发送给跟踪控制模块。

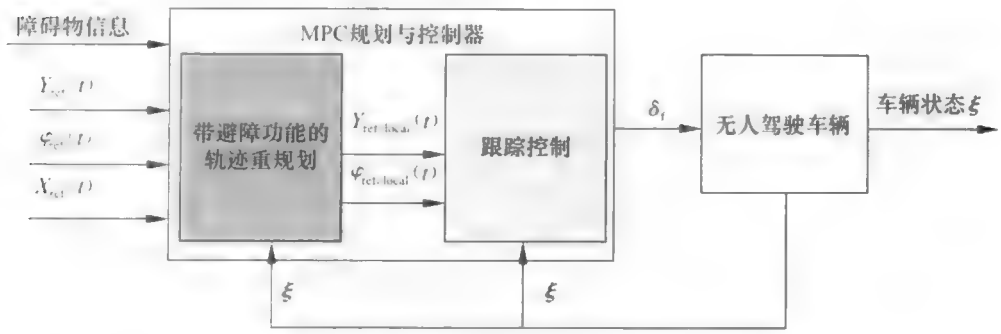


图 6.1 结合规划层的轨迹跟踪控制系统

基于 MPC 的轨迹规划层需要满足以下 3 个条件：

1 规划得到的期望轨迹与给定的参考路径之间的偏差要尽可能小。

2 规划得到的期望轨迹要满足车辆的运动学和动力学的约束条件。

3 规划得到的期望轨迹要能够避开障碍物，满足安全避障要求。

基于 MPC 的轨迹跟踪控制层需要满足以下 3 个条件：

1 实际的跟踪轨迹与期望轨迹之间的偏差要尽可能小。

2 优化计算得到的车辆控制量要满足执行机构的限制。

3 车辆行驶要满足安全性要求，这里指车辆行驶时不发生打滑或者侧翻。

在所构建的双层控制体系中都采用模型预测控制算法，因此需要确定两方面的问题：

(1) 规划层与控制层的模型选择

车辆模型的选择需要综合考虑性能表现以及计算量等因素。由于规划算法本身算法计算量大，而引入模型预测控制的主要目的是使规划结果要满足车辆动力学和运动学模型约束，因此为减少计算量，不宜采用太复杂的模型。根据文献[70]的对比实验可知，在规划层采用较低精度的模型而在控制层采用较高精度的模型，能够较好地兼顾控制性能和计算速度。因此，在规划层中采用忽略车身尺寸信息的点质量模型。

（2）控制周期选择

一般来说，规划层的规划周期可以比控制层的控制周期长。这主要是因为规划层一次可以规划出包含若干个控制周期的轨迹，而采用较短的规划周期对提升规划性能并没有帮助。在此，将规划层的规划周期取为控制层控制周期的两倍，经过后续仿真实验验证，该设定可以保证控制系统的实时性与可靠性。

6.2 基于 MPC 的轨迹规划器

轨迹重规划算法是轨迹规划层中的核心内容，主要工作是设计合理的评价函数，在满足各种约束条件下，实现避障功能并且尽量减小车辆与全局参考路径的偏差，最后通过合理的方式输出给跟踪控制层。

在规划层中采用点质量模型，如图 6.2 所示。根据第 2 章可以得到车辆的点质量模型，如式（6.1）和式（6.2）所示。

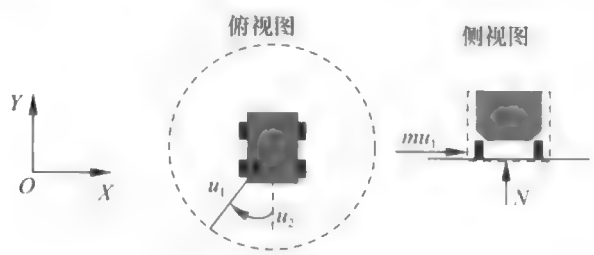


图 6.2 点质量模型示意图

$$\begin{aligned}
\ddot{y} &= a_y \\
\ddot{x} &= 0 \\
\dot{\varphi} &= \frac{a_y}{\dot{x}} \\
\dot{Y} &= \dot{x}\sin\varphi + \dot{y}\cos\varphi \\
\dot{x} &= x\cos\varphi - \dot{y}\sin\varphi
\end{aligned} \tag{6.1}$$

考虑车辆的动力学约束，加入如下约束条件：

$$|a_y| < ug \tag{6.2}$$

式（6.1）和式（6.2）可以简写成如下形式：

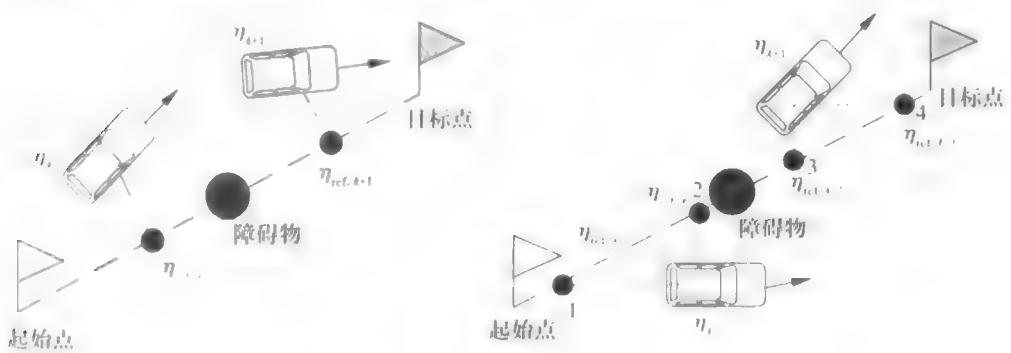
$$\begin{aligned}
\dot{\xi}(t) &= f(\xi(t), \mu(t)) \\
|\mu(t)| &< ug
\end{aligned} \tag{6.3}$$

式中，$\xi = [v, x, \phi, Y, V]^{\mathrm{T}}$，共有5个离散的状态变量，分别为车辆在$y$和$x$方向的车速、车辆航向角、车体位置的纵坐标和横坐标；u代表控制量，此处为车辆的前轮偏角δ。

6.2.1　参考点的选择

在轨迹重规划的时候需计算参考轨迹与预测轨迹之间的偏差$\boldsymbol{\eta} - \boldsymbol{\eta}_{\mathrm{ref}}$，而起始参考轨迹点如何选取则是其中的一个关键问题。Eklund等[80]利用几何法寻找全局参考轨迹上离车辆最近的轨迹点，如图6.3（a）所示。该方法利用道路的曲率信息，从车辆质心出发，在局部作垂直于道路的直线，直线与道路的交点即所寻找到的起始参考点。经过多次仿真实验发现，该方法对于尺寸较小的障碍物能够发挥作用，无人驾驶车辆可以在参考轨迹点的指引下顺利达到目标点；但如果障碍物尺寸较大，如杆状的障碍物，车辆的参考点在躲开障碍物后会出现重新回到起始点的情况。出现这个问题的原因在于没有将目标点信息纳入参考轨迹点寻找过程中，当车辆为了绕开障碍物而规划出航向角超过90°的轨迹时，车辆便会将已经走过的参考点重新作为新的参考点。

为解决该问题，本文将目标点信息加入到参考轨迹点的选择过程中。车辆在跟踪过程中无论处于任何位置，都会在全局坐标系下作出平行于x轴和y轴的两条直线与全局期望轨迹相交，如图6.3（b）所示。由于始终存在两个交点，故分别计算这两个交点与目标点的距离，选择距离目标点较近的参考点作为真正的参考起始点。因此，图中k时刻选择点2作为参考点，而在$k+1$时刻选择点3作为参考点。



（a）　　（b）

图6.3　寻找参考轨迹点的方法

（a）方法一；（b）方法二

6.2.2 避障功能函数

在无人驾驶车辆行驶环境中，障碍物信息通常由激光雷达的障碍物点给出。当这些障碍物点相互靠近时，可以通过聚类算法将它们形成一个大的障碍物，而障碍物点相互远离时，就可以将其看成多个离散的小障碍物。因此，避障功能函数可以针对每一个单独的障碍物进行设计。惩罚函数的基本思路是通过障碍物点与目标点的距离偏差来调节函数值的大小，且距离越近，函数值越大。综合车辆速度与目标函数中惩罚函数的比重对避让障碍物的影响，选择如下形式的避障功能函数：

$$J_{\mathrm{obs},i}=\frac{S_{\mathrm{obs}}v_i}{(x_i-x_0)^2+(y_i-y_0)^2+\zeta} \tag{6.4}$$

式中，S_{obs}为权重系数，$v_i=v_x^2+v_y^2$，$(x_i,\ y_i)$是障碍物点在车身坐标系下的位置坐标，$(x_0,\ y_0)$是车辆质心坐标，ζ为较小的正数，用于防止出现分母为0的现象。给定各参数后，函数值随障碍物相对坐标变化规律如图6.4（a）所示。

为了使轨迹规划的结果实际可行，需要根据车身尺寸对障碍物进行膨胀处理。本书参考文献［1］中的膨胀方法，以无人驾驶车辆运动中心的内切圆和外接圆半径为尺寸对障碍物膨胀。当障碍物尺寸较大时，还需要按照一定的分辨率对障碍物进行分割处理，防止出现车辆从障碍物中间穿越的现象，如图6.4（b）所示。

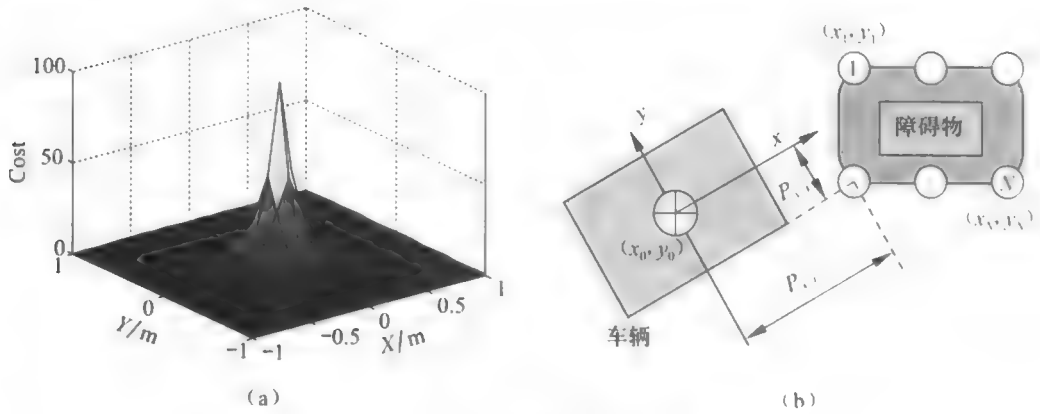


图6.4 惩罚函数与障碍物膨胀

（a）惩罚函数示意；（b）障碍物膨胀与分割

将上述避障功能函数代入式（6.4）中，设权重系数S_{obs}分别为10、100和1 000，给定不同初始状态后的规划结果如图6.5所示。当规划层接收到障

碍物信息后（图中点 1 和点 2），规划出避开障碍物的新轨迹，权重系数的增加会使规划结果趋于保守；而当车辆没有接收到障碍物信息（图中点 3）时，权重系数不会对规划结果产生任何影响。当车辆状态测量或者估计存在较大误差时，可以采用较大的权重系数，但同时也带来了跟踪偏差增加的问题。

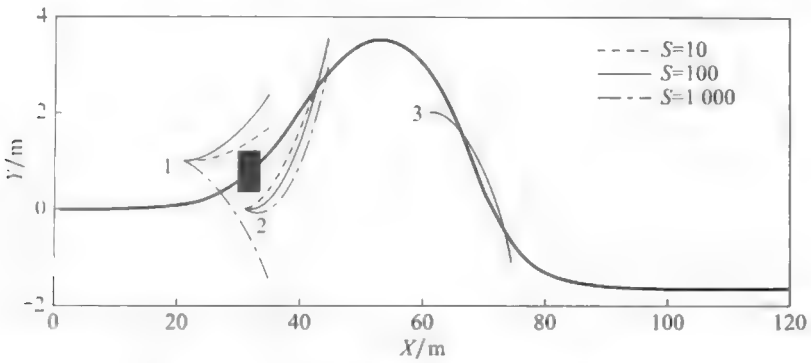


图 6.5　不同权重系数下的规划结果

轨迹重规划层中的控制目标是尽量减少与全局参考路径的偏差，并且实现对障碍物的避让。对障碍物的避让以惩罚函数的方式实现。轨迹重规划层的模型预测控制器具体形式如下：

$$\min_{\boldsymbol{U}_t} \sum_{i=1}^{N_p} \| \boldsymbol{\eta}(t+i \mid t) - \boldsymbol{\eta}_{\mathrm{ref}}(t+i \mid t) \|_{\boldsymbol{Q}}^2 + \| \boldsymbol{U}_i \|_{\boldsymbol{R}}^2 + J_{\mathrm{obs},i} \tag{6.5}$$

$$\text{s. t. } \boldsymbol{U}_{\min} \leqslant \boldsymbol{U}_i \leqslant \boldsymbol{U}_{\max}$$

式中，$J_{\mathrm{obs},i}$ 为采样时刻 i 的避障函数。

由于规划层的实时性要求比控制层低，点质量模型相对于非线性动力学模型也进行了较大程度的简化，具备更高求解精度的非线性模型预测控制算法完全能够满足轨迹重规划的要求，而非线性目标函数的采用也能给后续惩罚函数的设计带来便利。因此，对于式（6.5）不再进行线性化，而是直接基于非线性模型求解。相关解法已在第 3 章中介绍，此处不再重复。

6.2.3　5 次多项式轨迹拟合

在轨迹重规划算法中，将目标函数设定为有限时域内与参考点的距离偏差最小，所规划出的轨迹是以预测时域内离散的点给出的。随着预测时域的增加，这些局部参考轨迹点的数量也会随之增加。直接将这些点输入控制层，占用了过多的控制器输入接口，不利于控制器的规范化设计。另外，由于规划层与控制层的控制周期并不一致，控制层很难根据离散的参考轨迹点完成轨迹跟踪的任务。

综合上述因素，有必要对轨迹重规划算法所规划出的局部参考路径进行处理，实现规划层与控制层的顺利对接。曲线拟合是对离散点处理的主要方式，根据所采用曲线的不同，有样条曲线拟合、多项式拟合以及幂指数拟合等。由于车辆存在运动学约束，如车辆位置连续要求曲线是连续的，横摆角连续要求曲线是一阶连续，而加速度约束则要求曲线二阶连续。因此，本书选择5次多项式作为拟合曲线，形式如下：

$$\begin{aligned} Y &= a_0 t^5 + a_1 t^4 + a_2 t^3 + a_3 t^2 + a_4 t + a_5 \\ \varphi &= b_0 t^5 + b_1 t^4 + b_2 t^3 + b_3 t^2 + b_4 t + b_5 \end{aligned} \tag{6.6}$$

式中，$\boldsymbol{a}_p = [a_0, a_1, a_2, a_3, a_4, a_5]$，$\boldsymbol{b}_p = [b_0, b_1, b_2, b_3, b_4, b_5]$ 为待求参数。

设规划层预测时域 $N_p = 16$，拟合结果如图6.6所示。图6.6中，星号为轨迹重规划算法在预测时域内规划出的离散点，曲线为拟合后的轨迹曲线。

以拟合后各数据点残差的范数来表征拟合质量，对整个轨迹重规划过程中

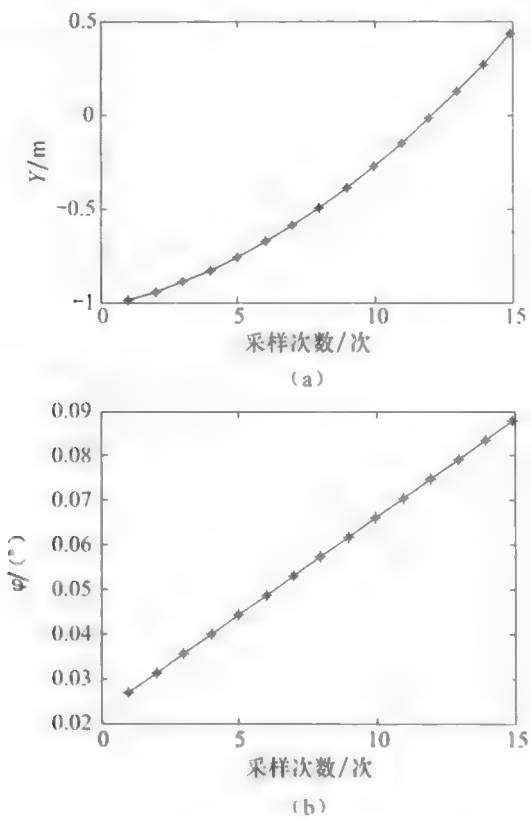


图6.6 离散轨迹点拟合结果

(a) 横向位置拟合结果；(b) 横摆角拟合结果

的201次规划数据进行统计，统计结果如图6.7所示。从图中可以看出，车辆位置横坐标 Y 的拟合残差最大值为0.003，车体航向角 φ 的拟合残差最大值不超过 7.50×10^{-5}，具备非常高的拟合精度。

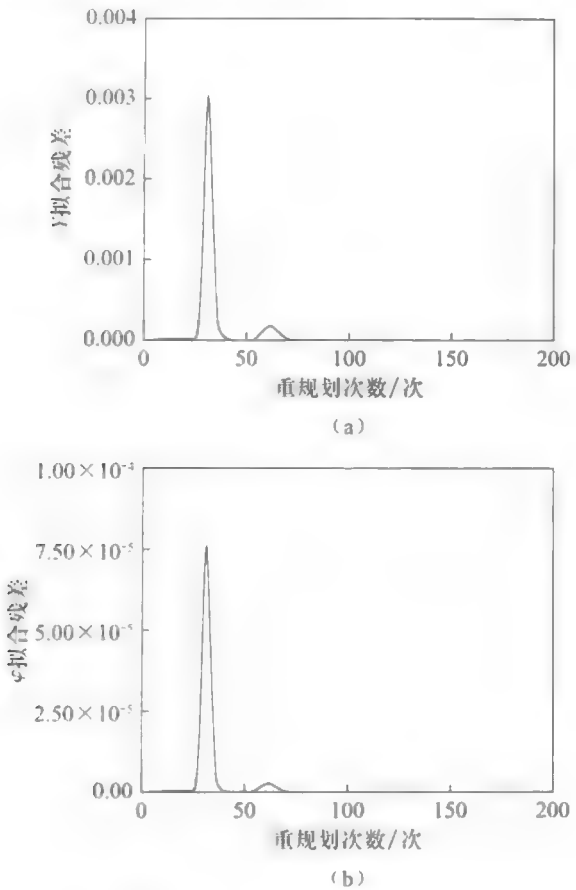


图6.7 数据点拟合质量统计

(a) 横向位置 Y 的拟合残差；(b) 横摆角

6.2.4 非线性二次规划计算

在用S函数实现基于MPC的轨迹规划器时首先需要确定S函数的输入与输出量，如图6.8所示。基于MPC的轨迹规划器将CarSim的输出作为输入，CarSim输出的参数为 v_x、v_y、ϕ、$\dot{\phi}$、Y、X，分别代表车辆在 Y 和 X 坐标方向上的速度、航向角及航向角速度、在 Y 和 X 坐标轴下的坐标等，共6个参数。基于MPC的轨迹规划器的输出是规划好的期望轨迹方程的参数，因为选

择了5次曲线进行曲线拟合，所以这里的输出量为10个

图6.8　各个模块的输入与输出示意

本节主要介绍如何通过Simulink的S-Function设计基于MPC的轨迹规划器。读者直接调用Chapter6_2_4.m文件生成S-Function形式的轨迹规划器。

为便于代码解析，将仿真程序文件Chapter6_2_4.m分为若干模块分别介绍。程序后面有备注，以方便读者阅读。

第1部分为主函数。通过此函数完成程序的初始化、并设置S函数模块的输入、输出和状态变量的个数等。sys变量在不同的阶段表示不同的意义。


```
function[sys,x0,str,ts]=MPC_TrajPlanner(t,x,u,flag)
% 该程序功能:用点质量模型设计规划期,能够规避障碍物
% 程序版本 V1.0,Matlab 版本:R2011a,采用 S 函数的标准形式,
% 状态量 = [y_dot,x_dot,phi,,Y,X],控制量为前轮偏角 ay
  switch flag,
  case 0      % flag = 0 表示处于初始化状态,此时用函数 mdlIni-
                tializeSizes 进行初始化
    [sys,x0,str,ts]=mdlInitializeSizes;% Initialization
  case 2          % flag = 2 表示此时要计算下一个离散状态
    sys = mdlUpdates(t,x,u);% Update discrete states
  case 3   % flag = 3 表示此时要计算输出
    sys = mdlOutputs(t,x,u);% Calculate outputs
  case {1,4,9}   % Unused flags
   % flag = 1 表示此时要计算连续状态的微分
   % flag = 4 表示此时要计算下一次采样的时间,主要用于变步长的设置
   % flag = 9 表示此时系统要结束
    sys = [];                  % 不使用的标志量置空
  otherwise
```

```
    error(['unhandled flag =',num2str(flag)]);
                                % Error handling
end
% End of dsfunc.
```

第2部分为初始化子函数。通过之前的描述可知，该模块有5个离散状态变量、6个输入变量、以及10个输出变量。其实现过程如下所示：

```
function[sys,x0,str,ts]=mdlInitializeSizes
    sizes = simsizes;                    % 用于设置模块参数的结构体
                                           用 simsizes 来生成
    sizes.NumContStates      =0;% 模块连续状态变量的个数
    sizes.NumDiscStates      =5;% 模块离散状态变量的个数
    sizes.NumOutputs         =10;% 输出期望的 Y 和 phi
    sizes.NumInputs          =6;% 模块输入变量的个数
    sizes.DirFeedthrough     =1;% 模块是否存在直接贯通
    sizes.NumSampleTimes     =1;% 模块的采样次数,至少是一个
    sys = simsizes(sizes);               % 设置完后赋给 sys 输出
    x0 = [0.001;0.0001;0.0001;0.00001;0.00001;];
                                         % 状态变量设置
    str = [];                            % 保留参数
    ts  = [0.1 0];                       % 采样周期:这里轨迹规划的周
                                           期设为100 ms
% End of mdlInitializeSizes
```

第3部分为更新离散状态量的子函数。

```
function sys = mdlUpdates(t,x,u)
sys = x;                                 % 更新状态变量
% End of mdlUpdate.
```

第4部分为计算输出的子函数，是基于MPC轨迹规划器的主体。

```
function sys = mdlOutputs(t,x,u)
% sys是CarSim的输出;t是时间变量;x是状态量,x=[y_dot,x_dot,
  phi,Y,X];
```

```
Nx = 5;                          % 状态量的个数,
Np = 15;                         % 预测步长
Nc = 2;                          % 控制步长
Nobs = 6;                        % 障碍物个数
T = 0.1;                         % Sample Time
y_dot = u(1)/3.6;                % 速度单位是 km/h,转换为 m/s
x_dot = u(2)/3.6 + 0.0001;% 单位是 km/h,转换为 m/s. 加一个很小的
                                   数可以防止分母为零
phi = u(3) * pi/180;             % CarSim 输出的为角度,角度转换为弧度
phi_dot = u(4) * pi/180;         % 角速度,角度转换为弧度
Y = u(5);                        % 单位为 m
X = u(6);                        % 单位为 m
% ================================================================
% 参考轨迹生成
% ================================================================
shape = 2.4;                     % 参数名称,用于参考轨迹生成
dx1 = 25;dx2 = 21.95;            % 只是参数名称
dy1 = 4.05;dy2 = 5.7;            % 只是参数名称
Xs1 = 27.19;Xs2 = 56.46;         % 参数名称
X_phi = 1: 1: 220;               % 这个点的区间是根据纵向速度(x_dot)
                                   来定的,
z1 = shape/dx1 * (X_phi - Xs1) - shape/2;
z2 = shape/dx2 * (X_phi - Xs2) - shape/2;
Y_ref = dy1/2. * (1 + tanh(z1)) - dy2/2. * (1 + tanh(z2));
% phi_ref = atan(dy1 * (1./cosh(z1)).^2 * (1.2/dx1) - dy2 *
  (1./cosh(z2)).^2 * (1.2/dx2));
% ================================================================
% 矩阵转换,将状态变量转化为状态变量矩阵
% ================================================================
State_Initial = zeros(Nx,1);
State_Initial(1,1) = y_dot;
State_Initial(2,1) = x_dot;
State_Initial(3,1) = phi;
State_Initial(4,1) = Y;
```

```
State_Initial(5,1)=X;
%% ------------------------------------------------------------------
%% 障碍物信息设置
%% ------------------------------------------------------------------
X_obstacle=zeros(Nobs,1);
X_obstacle(1:2)=30;
X_obstacle(3:4)=35;
X_obstacle(5:6)=32.5;
Y_obstacle=zeros(Nobs,1);
Y_obstacle(1)=0.5;
Y_obstacle(2)=1;
Y_obstacle(3)=0.5;
Y_obstacle(4)=1;
Y_obstacle(5)=0.5;
Y_obstacle(6)=1;
Yref=(Y_ref(1,round(State_Initial(5,1))+1:round(State_
Initial(5,1))+Np)';% Yref 采用的是近似算法,此处为局部期望路径
Q=1.*eye(Np,Np);  % 这里设置评价矩阵,都设为了1.可以根据跟
                    踪情况加以调整
R=20*eye(Nc,Nc);  %
S=100;            % 避障函数的权重
%% ------------------------------------------------------------------
%% 开始求解过程
%% ------------------------------------------------------------------
mu=0.4;           % 地面摩擦系数
g=9.8;
lb=[-mu*g;-mu*g];
ub=[mu*g;mu*g];   % 设置约束
A=[];
b=[];
Aeq=[];
beq=[];
options=optimset('Algorithm','active-set');
[A,fval,exitflag]=fmincon(@(x)MY_costfunction(x,State_
```

```
    Initial,Np,Nc,Nobs,T,Yref,Q,R,S,X_obstacle,Y_obsta-
cle),[0;0;],A,b,
    Aeq,beq,lb,ub,[],options);% 求解
fprintf('exitflag=%d\n',exitflag);
% ================================================================
% 计算输出
% ================================================================
y_dot_predict=zeros(Np,1);% 以下根据计算出的控制量推导所有
                            的状态量
x_dot_predict=zeros(Np,1);
phi_predict=zeros(Np,1);
Y_predict=zeros(Np,1);
X_predict=zeros(Np,1);
for i=1:1:Np
    if i==Nc-1
        ay(i)=A(1);
        y_dot_predict(i,1)=State_Initial(1,1)+T*ay
          (i);              % 以下完成状态量更新
        x_dot_predict(i,1)=State_Initial(2,1);
        phi_predict(i,1)=State_Initial(3,1)+T*ay(i)/
          State_Initial(2,1);
        Y_predict(i,1)=State_Initial(4,1)+T*(State_
                       Initial(2,1)*sin(State_Ini-
                       tial(3,1))+State_Initial(1,1)
                       *cos(State_Initial(3,1)));
        X_predict(i,1)=State_Initial(5,1)+T*(State_
                       Initial(2,1)*cos(State_Ini-
                       tial(3,1))-State_Initial(1,1)
                       *sin(State_Initial(3,1)));
    else
        ay(i)=A(2);% 这种写法是仅仅考虑两个控制周期
        y_dot_predict(i,1)=y_dot_predict(i-1,1)+T*
          ay(i);
        x_dot_predict(i,1)=State_Initial(2,1);
```

```
        phi_predict(i,1)=phi_predict(i-1,1)+T*ay(i)/
                         x_dot_predict(i-1,1);
        Y_predict(i,1)=Y_predict(i-1)+T*(State_Ini-
                       tial(2,1)*sin(phi_predict(i-
                       1))+y_dot_predict(i-1)*cos
                       (phi_predict(i-1)));
        X_predict(i,1)=X_predict(i-1)+T*(State_Ini-
                       tial(2,1)*cos(phi_predict(i-
                       1))-y_dot_predict(i-1)*sin
                       (phi_predict(i-1)));
    end
end
Paramater_X_Y=polyfit(X_predict,Y_predict,4);
Paramater_X_PHI=polyfit(X_predict,phi_predict,4);
OutPut(1:5)=Paramater_X_Y;
OutPut(6:10)=Paramater_X_PHI;
sys=OutPut;
% End of mdlOutputs.
```

第5部分为求代价函数的功能子函数。

```
function cost=MY_costfunction(x,State_Initial,Np,Nc,
              Nobs,T,,Yref,Q,R,S,Y_obstacle)   % 求代价
              函数的功能子函数
cost=0;
y_dot=State_Initial(1,1);
x_dot=State_Initial(2,1);
phi=State_Initial(3,1);
Y=State_Initial(4,1);
X_start=State_Initial(5,1);
y_dot_predict=zeros(Np,1);
x_dot_predict=zeros(Np,1);
phi_predict=zeros(Np,1);
Y_predict=zeros(Np,1);
```

```
X_predict = zeros(Np,1);
Y_error = zeros(Np,1);
J_obst = zeros(Np,1);
ay = zeros(Np,1);
for i = 1:1:Np
    if i == Nc - 1
        ay(i,1) = x(1);
        y_dot_predict(i,1) = y_dot + T * ay(i,1);
                        % 以下完成状态量更新
        x_dot_predict(i,1) = x_dot;
        phi_predict(i,1) = phi + T * ay(i) / x_dot;
        Y_predict(i,1) = Y + T * (x_dot * sin(phi) + y_dot *
                        cos(phi));
        X_predict(i,1) = X_start + T * (x_dot * cos(phi) - y_
                        dot * sin(phi));
        for j = 1:1:Nobs
            J_obst(i,1) = J_obst(i,1) + 1 / (((X_predict(i,1)) X
                        _obstacle(j,1))^2 + (Y_predict(i,
                        1) - Y_obstacle(j,1))^2 +
                        0.000001);
        end
    else
        ay(i,1) = x(2); % 这种写法是仅仅考虑两个控制周期
        y_dot_predict(i,1) = y_dot_predict(i-1,1) + T *
                        ay(i,1);
        x_dot_predict(i,1) = x_dot;
        phi_predict(i,1) = phi_predict(i-1,1) + T * ay(i) / x
                        _dot_predict(i-1,1);
        Y_predict(i,1) = Y_predict(i-1) + T * (x_dot * sin
                        (phi_predict(i-1)) + y_dot_pre-
                        dict(i-1) * cos(phi_predict(i-
                        1)));
        X_predict(i,1) = X_predict(i-1) + T * (x_dot * cos
                        (phi_predict(i-1)) - y_dot_pre-
```

```
                                dict(i-1)*sin(phi_predict(i-
                                1)));
            for p=1:1:Nobs
                J_obst(i,1)=J_obst(i,1)+1/(((X_predict(i,
                                1))-X_obstacle(p,1))^2 + (Y_
                                predict(i,1)-Y_obstacle(p,1))^
                                2+0.000001);
            end
        end
        Y_error(i,1)=Y_predict(i,1)-Yref(i,1);
                    % Yref 是局部期望路径,Y_ref 为全局期望路径
end
cost=cost+Y_error'*Q*Y_error+ay(1:2)'*R*ay(1:2)+S*
     sum(J_obst(:));
% End of CostFunction
```


6.3 基于 MPC 的路径跟踪控制器

为了减少计算量，这里选择车辆的线性时变模型进行 MPC 计算。模型状态量是：(Y、X、$\dot{\theta}$、θ、$\dot{Y}$、$\dot{X}$)，控制量为车辆的前轮偏角 δ。为保证控制的实时性，在跟踪控制层依然采用基于线性时变模型的模型预测控制算法，设计如下形式的跟踪控制器：

$$\min_{\Delta U_t} \sum_{i=1}^{N_p} \| \boldsymbol{\eta}(t+i|t) - \boldsymbol{\eta}_{ref,local}(t+i|t) \|_Q^2 + \sum_{i=1}^{N_c-1} \| \Delta \boldsymbol{u}(t+i|t) \|_R^2 + \rho \varepsilon^2 \tag{6.7}$$

$$\text{s.t. } \Delta \boldsymbol{U}_{min} \leqslant \Delta \boldsymbol{U}_t \leqslant \Delta \boldsymbol{U}_{max}$$

$$\boldsymbol{U}_{min} \leqslant \boldsymbol{A} \Delta \boldsymbol{U}_t + \boldsymbol{U}_t \leqslant \boldsymbol{U}_{max}$$

$$\boldsymbol{y}_{lc,min} \leqslant \boldsymbol{y}_{lc} \leqslant \boldsymbol{y}_{lc,max}$$

$$\boldsymbol{y}_{sc,min} - \varepsilon \leqslant \boldsymbol{y}_{sc} \leqslant \boldsymbol{y}_{lc,max} + \varepsilon$$

$$\varepsilon > 0$$

式中，$\boldsymbol{\eta}_{ref,local} = [Y_{ref,local}, \varphi_{ref,local}]$ 为轨迹重规划层输出的局部参考轨迹。

本章的轨迹跟踪控制器是基于车辆简化动力学模型（小角度假设）设计

的，在第5章有详细介绍，这里不再赘述。本节主要介绍如何通过Simulink的S-Function设计基于MPC的轨迹跟踪控制器，可直接调用Chapter6_3.m文件生成S-Function形式的轨迹规划器。

为便于代码解析，将该仿真程序分为若干模块分别介绍，程序后面有备注，以方便读者阅读。

Chapter6_3.m第1部分为主函数。

```
function[sys,x0,str,ts]=MPC_Controller(t,x,u,flag)
% 该程序功能:用LTV MPC和车辆简化动力学模型(小角度假设)设计控制
  器,接受来自规划器的期望轨迹,实现轨迹跟踪控制功能
% 程序版本V1.0,Matlab版本:R2011a,采用S函数的标准形式,
% 状态量=[y_dot,x_dot,phi,phi_dot,Y,X],控制量为前轮偏角
  delta_f
switch flag,
  case 0
    [sys,x0,str,ts]=mdlInitializeSizes;% Initialization
  case 2
    sys=mdlUpdates(t,x,u);% Update discrete states
  case 3
    sys=mdlOutputs(t,x,u);% Calculate outputs
  case{1,4,9}% Unused flags
    sys=[];
  otherwise
    error([ünhandled flag=',num2str(flag)]);% Error
      handling
end
% End of dsfunc.
```

Chapter6_3.m第2部分为初始化子函数。

```
function[sys,x0,str,ts]=mdlInitializeSizes
sizes=simsizes;
sizes.NumContStates  =0;%
sizes.NumDiscStates  =6;%
```

```
sizes.NumOutputs      =1;
sizes.NumInputs       =16;
sizes.DirFeedthrough  =1;%
sizes.NumSampleTimes  =1;
sys=simsizes(sizes);
x0=[0.001;0.0001;0.0001;0.00001;0.00001;0.00001];
global U;
U=[0];                    % 控制量初始化
str=[];               % Set str to an empty matrix.
ts=[0.02 0];        % sample time:[period,offset]
% End of mdlInitializeSizes
```

Chapter6_2 MPC_Controller.m 第3部分为更新离散状态量子函数：

```
function sys=mdlUpdates(t,x,u)
sys=x;
% End of mdlUpdate.
```

第4部分为计算输出子函数，是基于MPC轨迹规划器的主体，

```
function sys=mdlOutputs(t,x,u)
global a b;
global U;
Nx=6;       % 状态量的个数
Nu=1;       % 控制量的个数
Ny=2;       % 输出量的个数，这边可以先用两个输出量
Np=25;      % 预测步长
Nc=10;      % 控制步长
Row=1000;  % 松弛因子
fprintf('Update start,t=%6.3f\n',t)
y_dot=u(1)/3.6;% 单位是km/h,转换为m/s
x_dot=u(2)/3.6+0.0001;% 后面加一个非常小的数，是防止出现分母为零的情况
phi=u(3)*3.14159265/180;% CarSim输出的为角度，角度转换为弧度
```

```
phi_dot = u(4) * 3.141592654 /180;
Y = u(5);% 单位为 m
X = u(6);% 单位为 m
% ================================================================
% 根据规划器的输入确定参考轨迹
% ================================================================
Paramater_X_Y(1,1) = u(7);% 读入局部规划器的数据,传递进来的是 5 次曲线的参数
Paramater_X_Y(1,2) = u(8);
Paramater_X_Y(1,3) = u(9);
Paramater_X_Y(1,4) = u(10);
Paramater_X_Y(1,5) = u(11);
Paramater_X_phi(1,1) = u(12);
Paramater_X_phi(1,2) = u(13);
Paramater_X_phi(1,3) = u(14);
Paramater_X_phi(1,4) = u(15);
Paramater_X_phi(1,5) = u(16);
% ================================================================
% 车辆参数输入
% ================================================================
Sf = 0.2;Sr = 0.2;% syms sf sr;% 分别为前后车轮的滑移率
lf = 1.232;lr = 1.468;% syms lf lr;% 前后车轮距离车辆质心的距离,车辆固有参数
Ccf = 66900;Ccr = 62700;Clf = 66900;Clr = 62700;% 分别为前后车轮的纵横向侧偏刚度
m = 1723;g = 9.8;I = 4175;% m 为车辆质量,g 为重力加速度,I 为车辆绕 Z 轴的转动惯量
% ================================================================
% 参考轨迹生成
% ================================================================
X_predict = zeros(Np,1);% 用于保存预测时域内的纵向位置信息,是计算期望轨迹的基础
phi_ref = zeros(Np,1);% 用于保存预测时域内的期望轨迹
```

```
Y_ref = zeros(Np,1);% 用于保存预测时域内的期望轨迹
dphi_ref = zeros(Np,1);
%% ------------------------------------------------------------------
%% 矩阵转换
%% ------------------------------------------------------------------
kesi = zeros(Nx + Nu,1);
kesi(1) = y_dot;% u(1) == X(1)
kesi(2) = x_dot;% u(2) == X(2)
kesi(3) = phi;% u(3) == X(3)
kesi(4) = phi_dot;
kesi(5) = Y;
kesi(6) = X;
[A_cell,B_cell,C] = [illegible];% 计算 kesi,即状态量与控制量合在一起
delta_f = U(1);
fprintf('Update start,u(1) = %4.2f\n',U(1))
T = 0.02;% 仿真步长
T_all = 20;% 临时设定,总的仿真时间,主要功能是防止计算期望轨迹越界
%% ------------------------------------------------------------------
%% 矩阵设置
%% ------------------------------------------------------------------
Q_cell = cell(Np,Np);% 权重设置
for i = 1:1:Np;
    for j = 1:1:Np;
        if i == j
            Q_cell{i,j} = [2000 0;0 10000;];
        else
            Q_cell{i,j} = zeros(Ny,Ny);
        end
    end
end
R = 5 * 10^5 * eye(Nu * Nc);
% 最基本也最重要的矩阵,是控制器的基础,采用动力学模型,该矩阵与车
  辆参数密切相关,通过对动力学方程求解雅克比矩阵得到
a = [1 - (259200 * T)/(1723 * x_dot), - T * (phi_dot + (2 *
```

```
    ((460218 * phi_dot)/5 - 62700 * y_dot))/(1723 * x_dot^2)-
    (133800 * ((154 * phi_dot)/125 + y_dot))/(1723 * x_dot^
    2)),0, - T * (x_dot - 96228/(8615 * x_dot)),0,0
    T * (phi_dot - (133800 * delta_f)/(1723 * x_dot)),
    (133800 * T * delta_f * ((154 * phi_dot)/125 + y_dot))/
    (1723 * x_dot^2)+1,
    0,T * (y_dot - (824208 * delta_f)/(8615 * x_dot)),0,0
    0,0,1,T,0,0
   (33063689036759 * T)/(7172595384320 * x_dot), T *
   (((2321344006605451863 * phi_dot)/8589934592000 -
   (6325188028897689 * y_dot)/34359738368)/(4175 * x_dot^
   2)+(5663914248162509 * ((154 * phi_dot)/125 + y_dot))/
   (143451907686400 * x_dot^2)),
    0,1 - (813165919007900927 * T)/(7172595384320000 * x_
      dot),0,0
    T * cos(phi),T * sin(phi),  T * (x_dot * cos(phi) - y_dot *
      sin(phi)),0,1,0
   - T * sin(phi),T * cos(phi), - T * (y_dot * cos(phi) + x_dot *
      sin(phi)),0,0,1];
b = [133800 * T/1723;T * ((267600 * delta_f)/1723 - (133800 *
    ((154 * phi_dot)/125 + y_dot))/(1723 * x_dot)); 0;
    5663914248162509 * T/143451907686400;0;0];
d_k = zeros(Nx,1);% 计算偏差,即 falcone42 页中的 d(k,t)
state_k1 = zeros(Nx,1);% 预测的下一时刻状态量,用于计算偏差,
% 以下为根据离散非线性模型预测下一时刻状态量.% 注意,为避免前后轴
  距的表达式(a,b)与控制器的 a,b 矩阵冲突,将前后轴距的表达式改为
  lf 和 lr.
state_k1(1,1) = y_dot + T * ( - x_dot * phi_dot + 2 * (Ccf * (del-
                ta_f - (y_dot + lf * phi_dot)/x_dot) + Ccr * (lr
                * phi_dot - y_dot)/x_dot)/m);
state_k1(2,1) = x_dot + T * (y_dot * phi_dot + 2 * (Clf * Sf + Clr
                * Sr + Ccf * delta_f * (delta_f - (y_dot + phi_
                dot * lf)/x_dot))/m);
state_k1(3,1) = phi + T * phi_dot;
```

```
state_k1(4,1)=phi_dot+T*((2*lf*Ccf*(delta_f-(y_dot+
              lf*phi_dot)/x_dot)-2*lr*Ccr*(lr*phi_
              dot-y_dot)/x_dot)/I);
state_k1(5,1)=Y+T*(x_dot*sin(phi)+y_dot*cos(phi));
state_k1(6,1)=X+T*(x_dot*cos(phi)-y_dot*sin(phi));
d_k=state_k1-a*kesi(1:6,1)-b*kesi(7,1);
d_piao_k=zeros(Nx+Nu,1);% d_k 的增广形式
d_piao_k(1:6,1)=d_k;
d_piao_k(7,1)=0;
A_cell=cell(2,2);
B_cell=cell(2,1);
A_cell{1,1}=a;
A_cell{1,2}=b;
A_cell{2,1}=zeros(Nu,Nx);
A_cell{2,2}=eye(Nu);
B_cell{1,1}=b;
B_cell{2,1}=eye(Nu);
A=cell2mat(A_cell);
B=cell2mat(B_cell);
C=[0 0 1 0 0 0 0;0 0 0 0 1 0 0;];
PSI_cell=cell(Np,1);%
THETA_cell=cell(Np,Nc);
GAMMA_cell=cell(Np,Np);%
PHI_cell=cell(Np,1);%
for p=1:1:Np;
    PHI_cell{p,1}=d_piao_k;% 为了简便,这里认为近似相等
    for q=1:1:Np;
        if q<=p;
            GAMMA_cell{p,q}=C*A^(p-q);
        else
            GAMMA_cell{p,q}=zeros(Ny,Nx+Nu);
        end
    end
end
```

```
for j=1:1:Np
    PSI_cell{j,1}=C*A^j;
    for k=1:1:Nc
        if k<=j
            THETA_cell{j,k}=C*A^(j-k)*B;
        else
            THETA_cell{j,k}=zeros(Ny,Nu);
        end
    end
end
PSI=cell2mat(PSI_cell);% size(PSI)=[Ny*Np Nx+Nu]
THETA=cell2mat(THETA_cell);% size(THETA)=[Ny*Np Nu*
      Nc]
GAMMA=cell2mat(GAMMA_cell);
PHI=cell2mat(PHI_cell);
Q=cell2mat(Q_cell);
H_cell=cell(2,2);
H_cell{1,1}=THETA'*Q*THETA+R;
H_cell{1,2}=zeros(Nu*Nc,1);
H_cell{2,1}=zeros(1,Nu*Nc);
H_cell{2,2}=Row;
H=cell2mat(H_cell);
error_1=zeros(Ny*Np,1);
Yita_ref_cell=cell(Np,1);
for p=1:1:Np
    if t+p*T>T_all
        X_DOT=x_dot*cos(phi)-y_dot*sin(phi);% 惯性坐标系
                                              下纵向速度
        X_predict(Np,1)=X+X_DOT*Np*T;
        Y_ref(p,1)=polyval(Paramater_X_Y,X_predict(Np,1));
        phi_ref(p,1)=polyval(Paramater_X_phi,X_predict
          (Np,1));
        Yita_ref_cell{p,1}=[phi_ref(p,1);Y_ref(p,1)];
    else
```

```
        X_DOT = x_dot * cos(phi)-y_dot * sin(phi);% 惯性坐标系
                                                  下纵向速度
        X_predict(p,1)= X + X_DOT * p * T;% 首先计算出未来 X 的位
                                            置,X(t)= X + X_dot *
                                            t
        Y_ref(p,1)=polyval(Paramater_X_Y,X_predict(p,1));
        phi_ref(p,1)= polyval(Paramater_X_phi,X_predict
                    (p,1));
        Yita_ref_cell{p,1}= [phi_ref(p,1);Y_ref(p,1)];
    end
end
Yita_ref = cell2mat(Yita_ref_cell);
error_1 = Yita_ref - PSI * kesi - GAMMA * PHI;% 求偏差
f_cell = cell(1,2);
f_cell{1,1}=2 * error_1' * Q * THETA;
f_cell{1,2}=0;
f = -cell2mat(f_cell);
%% ============================================
% 控制量约束
%% ============================================
A_t = zeros(Nc,Nc);
for p = 1:1:Nc% % 以下为约束生成区域
    for q = 1:1:Nc
        if q <= p
            A_t(p,q)=1;
        else
            A_t(p,q)=0;
        end
    end
end
A_I = kron(A_t,eye(Nu));
Ut = kron(ones(Nc,1),U(1));
umin = -0.1744;% 维数与控制变量的个数相同
umax = 0.1744;
```

```
delta_umin = -0.0148 * 0.4;
delta_umax = 0.0148 * 0.4;
Umin = kron(ones(Nc,1),umin);
Umax = kron(ones(Nc,1),umax);
ycmax = [0.21;5];
ycmin = [-0.3;-3];% 输出量约束(硬约束设置),此处让所有输出为硬
                    约束
Ycmax = kron(ones(Np,1),ycmax);
Ycmin = kron(ones(Np,1),ycmin);
% 结合控制量约束和输出量约束
A_cons_cell = {A_I zeros(Nu * Nc,1); -A_I zeros(Nu * Nc,1);
  THETA zeros(Ny * Np,1); -THETA zeros(Ny * Np,1)};
b_cons_cell = {Umax - Ut; -Umin + Ut;Ycmax - PSI * kesi - GAMMA *
  PHI; -Ycmin + PSI * kesi + GAMMA * PHI};
A_cons = cell2mat(A_cons_cell);% 状态量不等式约束增益矩阵,转
                                 换为绝对值的取值范围
b_cons = cell2mat(b_cons_cell);% (求解方程)状态量不等式约束
                                 的取值
M = 10;
delta_Umin = kron(ones(Nc,1),delta_umin);
delta_Umax = kron(ones(Nc,1),delta_umax);% 状态量约束
lb = [delta_Umin;0];% (求解方程)状态量下界,包含控制时域内控制
                      增量和松弛因子
ub = [delta_Umax;M];% (求解方程)状态量上界,包含控制时域内控制
                      增量和松弛因子
% ================================================================
% 开始求解过程
% ================================================================
options = optimset('Algorithm','active-set');
x_start = zeros(Nc+1,1);% 加入一个起始点
[X,fval,exitflag] = quadprog(H,f,A_cons,b_cons,[],[],lb,
                    ub,x_start,options);
fprintf('exitflag = %d\n',exitflag);
fprintf('H = %4.2f\n',H(1,1));
```

```
fprintf('f = %4.2f\n',f(1,1));
%  ================================================================
% 计算输出
%  ================================================================
u_piao = X(1);% 得到控制增量
U(1) = kesi(7,1) + u_piao;% 当前时刻的控制量为上一时刻控制+控制增量
sys = U;
% End of mdlOutputs.
```

6.4 不同车速下的跟踪控制仿真实例验证

设计仿真工况，测试轨迹跟踪控制系统在不同速度下的跟踪能力。依然以双移线轨迹作为车辆的全局参考轨迹，仿真工况设定如下：

轨迹跟踪仿真实验分别在 10 m/s、20 m/s 和 30 m/s 的速度下进行，道路附着条件良好，$\mu = 0.8$。参考轨迹上存在一个障碍物，膨胀后的尺寸为 5m × 2 m，角点的坐标位置为（30，0.5）。上层规划器所采用的参数为：$T = 0.1$ s，$N_p = 15$，$N_c = 2$，$Q = 10$，$R = 5$，$S_{obs} = 10$。下层跟踪控制器所采用的参数为：$T = 0.05$ s，$N_p = 25$，$N_c = 10$，$-10° \leqslant \delta_f \leqslant 10°$，$-0.85° \leqslant \Delta\delta_f \leqslant 0.85°$，权重矩阵设置为：

$$\boldsymbol{Q} = \begin{bmatrix} 200 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{bmatrix}, \quad \boldsymbol{R} = 1.1 \times 10^5, \quad \rho = 1\,000$$

下面介绍 CarSim/Simulink 进行联合仿真的具体步骤：

首先要新建一组 Datasets，出现如图 6.9 所示的对话框。在两个文本框中分别输入“Example”和“MPCTracker”，然后点击“Set”，完成新建。此时，选择主菜单中的“Datasets”下拉菜单，发现“Example”一样中多出“MPC-Tracker”

图 6.9 新建 Dataset

然后进入模型编辑界面，如图6.10所示。

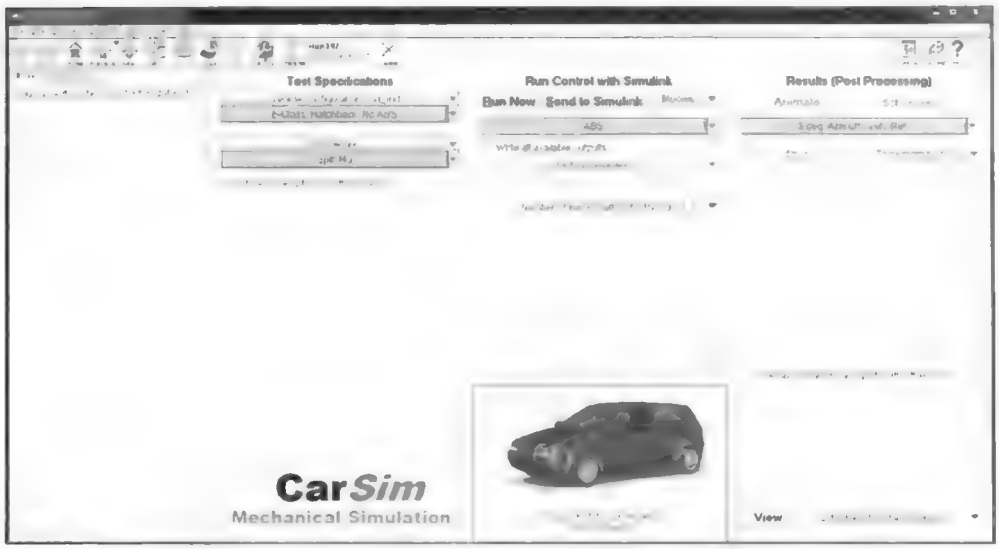


图6.10　CarSim主界面

在此界面进行车辆参数及仿真工况的设置、数学模型求解以及后处理。下面进行详细介绍。

6.4.1　车辆参数设置

在车辆参数设置部分，需要设置的车辆参数有整车尺寸和外形、系统和前后悬架等，如图6.11所示。

按照德国汽车分级标准，A级（包括A，A0，A00）车是指小型轿车，B级车是中档轿车，C级车是高档轿车，而D级车指的则是豪华轿车。其等级划分主要依据轴距、排量、重量等参数，字母顺序越靠后，该级别车的轴距越长，排量和重量越大，轿车的豪华程度也不断提高。

这里我们以比亚迪速锐汽车作为原型进行建模。速锐排量1.5 L，轴距2.66 m，车身长度4.68 m。这里在C级车的基础上进行修改。点击图6.12中的下三角，选择相应的车型。这里选择“CS C-Class，Hatchback”车型。

6.4.2　仿真工况设置

首先进入仿真工况设置界面，如图6.13所示。

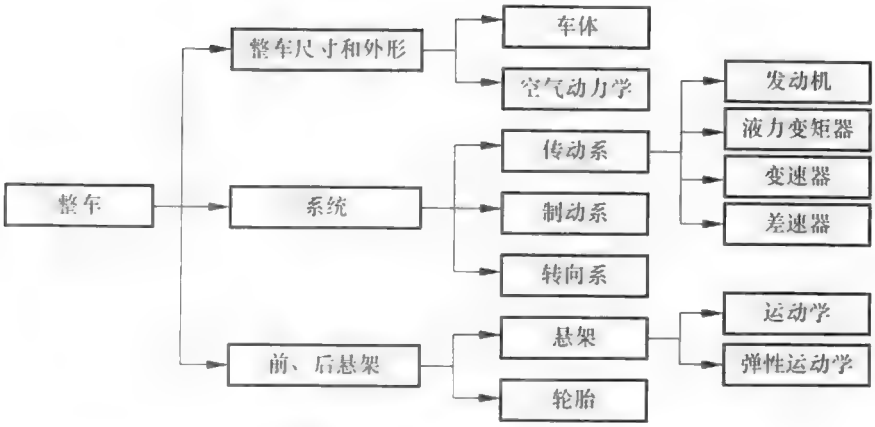


图 6.11 车辆参数设置内容

（1）仿真工况参数设置

点击“New”，新建仿真工况，在图 6.14 所示的文本框中依次输入“MPC Example”和“MPC ENV”，点击“Set”，

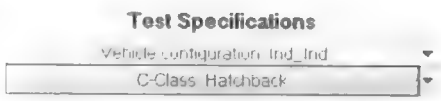


图 6.12 选择车型

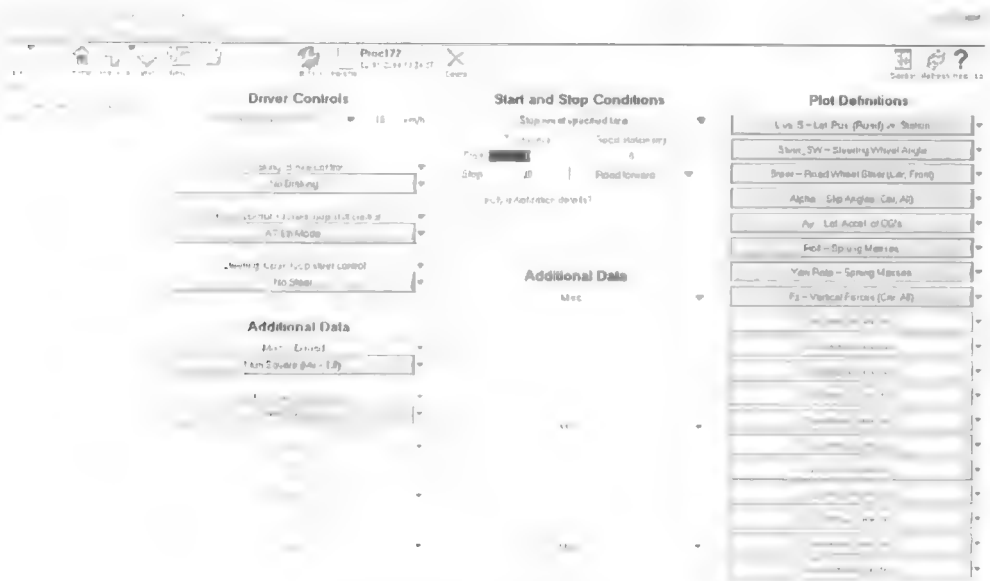


图 6.13 仿真工况的主界面

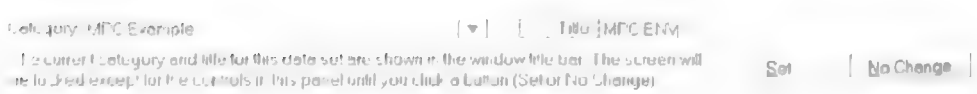


图 6.14 新建仿真工况

完成新建。

根据要求设置仿真工况：

① 目标车速为 18 km/h。

② 无制动。

③ 挡位控制选用闭环 AT 5 挡模式。

④ 无转向控制，方向盘转角为 0 deg。

⑤ 路面选择为 1 km^2，摩擦系数为 1.0 的方形路面

⑥ 完成设置后，如图 6.15 所示。

（2）设置仿真时间

同样在设置仿真工况的主界面，在如图 6.16 所示的文本框内输入“100”设置完成后，点击“Save”，进行保存。

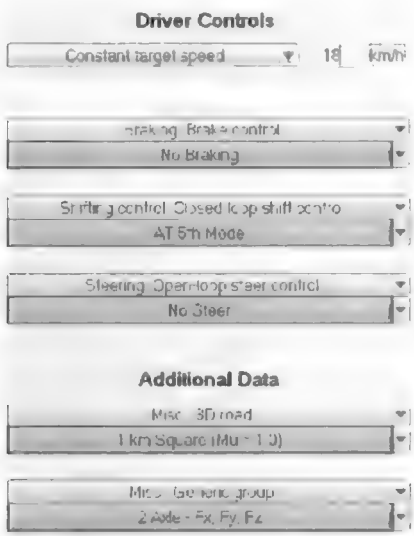


图 6.15　仿真工况设置

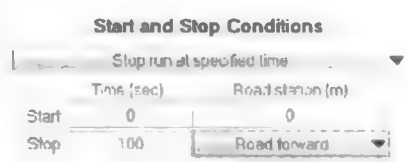


图 6.16　仿真时间设置

（3）设置仿真步长

在 CarSim 主界面中，点击下拉菜单“Tools”，选择“Preference”，出现如图 6.17 所示的界面，将仿真步长设置为“0.001”。

点击“Home”图标，返回 CarSim 的主界面，在“Procedure”项目下选择前面新建的“MPC ENV”工况，完成仿真工况设置，如图 6.18 所示

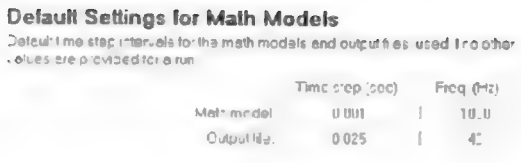


图 6.17 设置仿真步长

图 6.18 新建仿真工况

6.4.3 CarSim/Simulink 联合求解

(1) 建立 CarSim/Simulink 的关联

点击如图 6.19 所示的 Models 选项，选择“Models: Simulink”。

点击如图 6.20 所示的选项，选择“[Link to New Dataset]”。

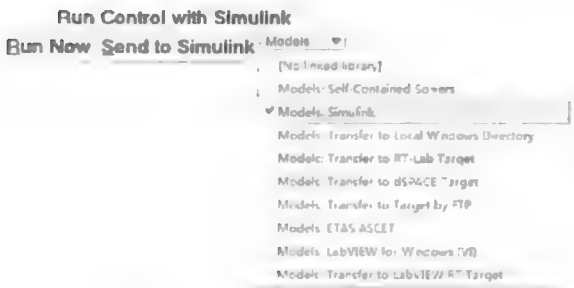


图 6.19 选择 Simulink 接口

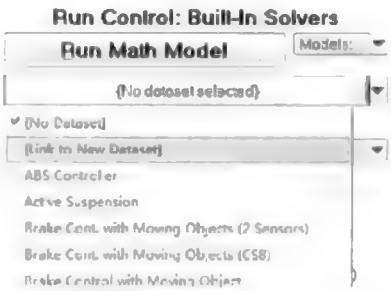


图 6.20 Link to New Dataset

此时将弹出一个如图 6.21 所示的对话框，在对话框中依次输入“Example”和“MPC_Simulink”，点击“Create and Link”完成新建

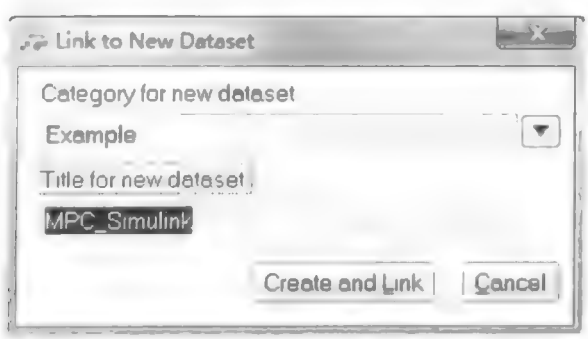


图 6.21 新建一个与 Simulink 联合的 Dataset

创建完新的 Dataset 之后，将其与 Simulink 关联起来，如图 6.22 所示

在将 CarSim 的模型关联到 Simulink 之前，需要先新建一个 Simulink 文件并保存。点击如图 6.23 所示的“MPC_ Simulink”，将会弹出如图 6.24 所示的 Simulink | Example | MPC_ Simulink 的主界面。

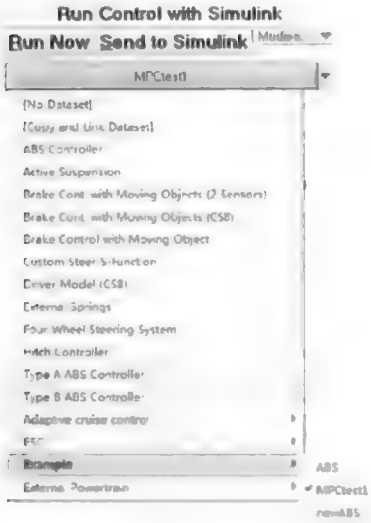


图 6.22　选择新建的“MPCtest1” dataset

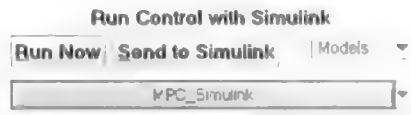


图 6.23　点击“MPC_ Simulink”

图 6.24　MPC_ Simulink 的主界面

选择相关路径，这里的工作路径和 Simulink Model 的路径是指 Simulink 文件保存的路径，可以根据实际情况选择“Use 64 - bit matlab”或者“Use 32 - bit matlab”，如图 6.25 所示。

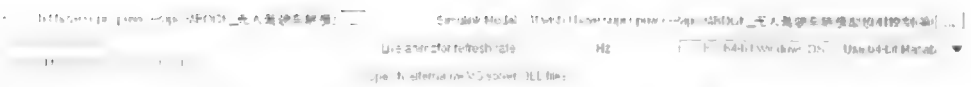


图 6.25 CarSim 模型导入 Simulink 路径设置

（2）定义 CarSim 的导入变量

点击如图 6.26（a）所示的“Import Channels”，选择“I/O Channels: Import”，然后点击如图 6.26（b）所示的“Copy and Link Dataset”，出现如图 6.26（c）所示的对话框，输入“MPC_ Simulink Input”。

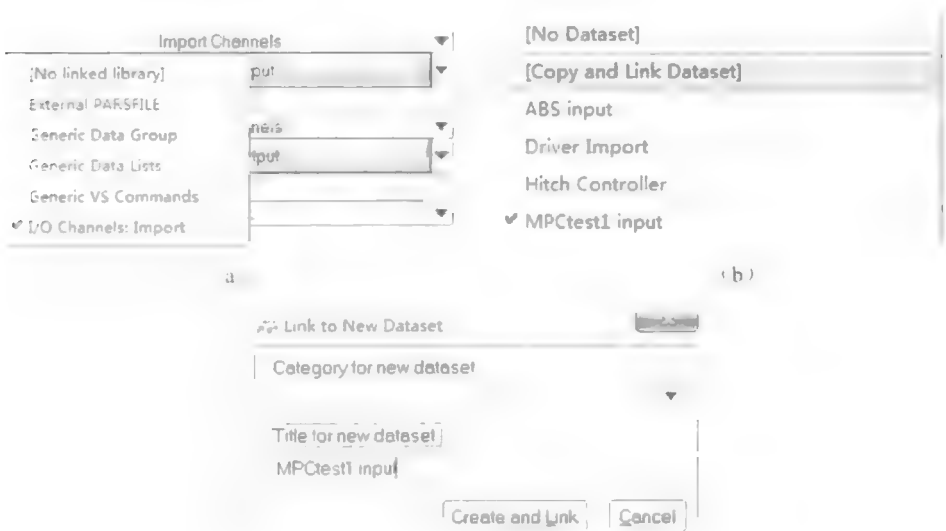


图 6.26 新建 MPC_ Simulink Input

点击“MPC_ Simulink Input”，显示如图 6.27 所示的界面

这里需要通过浏览找到“Readme file imports:”，其后显示为：

Programs \ solvers \ ReadMe \ f_ i_ i_ _ s_ imports_ tab. txt

定义 CarSim 的导入变量为车轮的前轮偏角，顺序依次为 IMP_ STEER_ F1（左前轮转角 deg），IMP_ STEER_ R1（右前轮转角 deg），IMP_ STEER_ L2（左后轮转角 deg），以及 IMP_ STEER_ R2（右后轮转角 deg）。需要注意的是：CarSim 的导入变量和 Simulink 中 MPC 模型的输出量是相对应的，所以这里的 CarSim 的导入变量应按照如图 6.28 所示的顺序排列选择

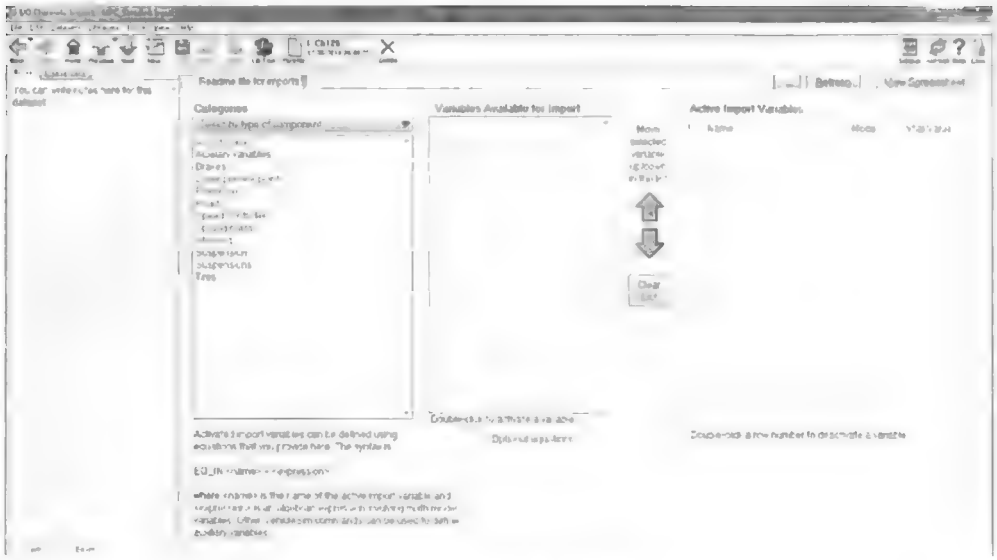


图 6.27　导入变量设置界面

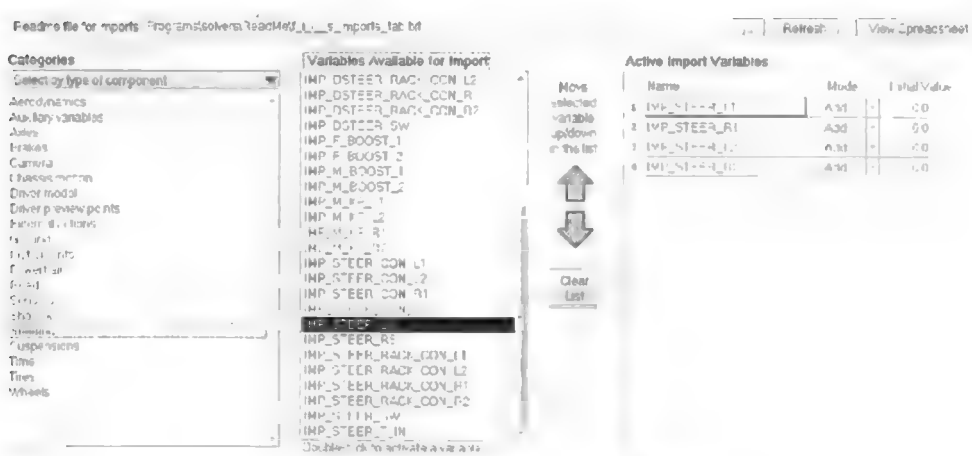


图 6.28　定义 CarSim 导入变量

(3) 定义 CarSim 的导出变量

步骤与本节 (2) 定义 CarSim 导入变量相同，首先新建名为“MPC_ Simulink Output”的一个 Dataset。点击“MPC_ Simulink Output”，弹出如图 6.29 所示的界面。

通过浏览找到 Readme file for outputs，其后显示为：

Programs \ solvers \ ReadMe \ i_ i_ outputs_ tab. txt

定义 CarSim 的导出变量依次为 VyBf_ SM（质心处的纵向车速 [km/h]），

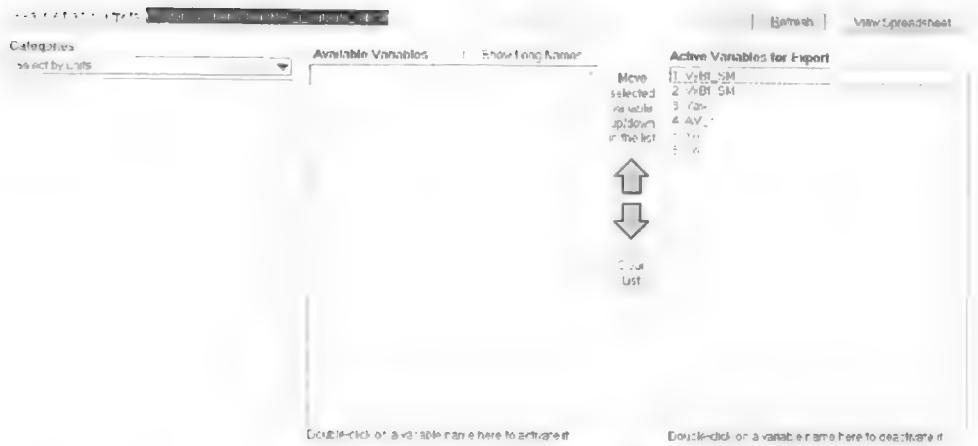


图 6.29 定义 CarSim 导出变量

VxBf_ SM、Yaw（偏航角［deg］），AV_ Y，Y0（坐标系 Y 轴的坐标值 m）和 X0（坐标系 X 轴的坐标值［m］）。需要注意的是：CarSim 的导出变量和 Simulink 中 MPC 模型的输入量是相对应的，所以这里的 CarSim 的导出变量应按照如图 6.30 所示的顺序排列选择。

（4）将 CarSim 建立的车辆模型输出到 Simulink

点击“Home”，返回 CarSim 的主界面，点击“Send to Simulink”，出现如图 6.31 所示的界面，此时 Matlab 及之前新建的空白模型“Chapter6_ 4_ 3. mdl”将被打开。

图 6.30 CarSim 的导出变量

图 6.31 数学模型求解器

如图 6.32 所示，在 Matlab Command Window 中输入“simulink”，回车，打开“Simulink Library Browser”。注意：现在的 Simulink Library Browser 比单独运算 Matlab/Simulink 时多了一个“CarSim S－Function”，如图 6.33 所示。

将“CarSim S－Function”拖曳到之前新建的 simulink 文件中，注意此模块恰好有一个输入接口和一个输出接口，分别对应 CarSim 的导入变量和导出变量。

完成上述设定后，CarSim 通过外部接口将车辆模型发送至指定路径下的

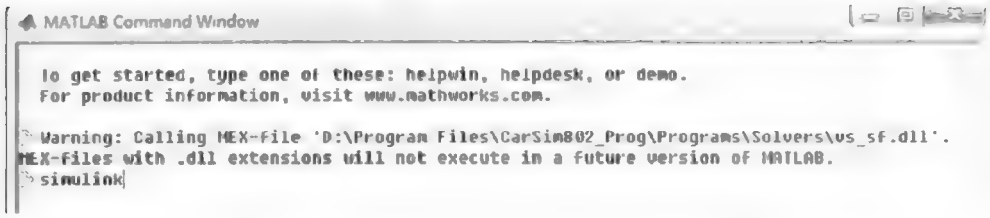


图 6.32　Matlab Command Window 命令窗口

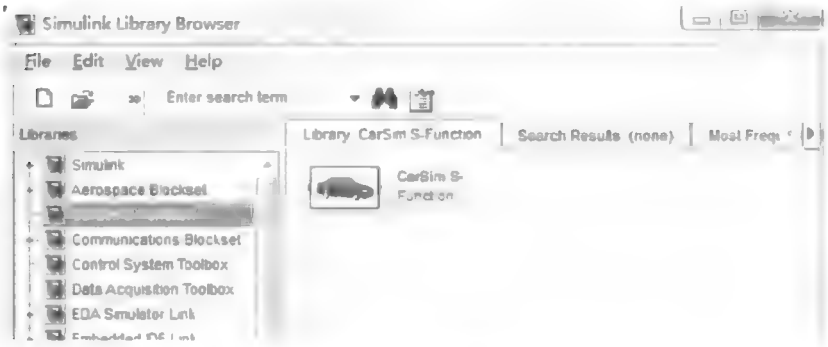


图 6.33　含有 CarSim S－Function 模块的 Simulink Library Browser

Simulink 仿真文件中，CarSim 模块即以 S－Function 的形式增加到 Simulink 模型库中。

（5）*S* 函数与 Simulink 环境下的仿真

在 Chapter6_ 4_ 3. mdl 中加入基于 MPC 的轨迹跟踪控制器，CarSim 的导出量经过该控制器的计算，决策出下一时刻的质心车速和前轮偏角，然后导入到 CarSim 模块里。

将 Simulink Library Browser 中的“S－Function”拖曳到 simulink 文件中，双击该图标，弹出如图 6.34 所示的对话框。在“S－function name”中输入“chapter6_ 2_ 4. m”，点击“OK”，将设计好的基于 MPC 的轨迹跟踪控制算法的 m 文件导入 simulink 文件。注意此模块恰好有一个输入接口和一个输出接口，分别对应着轨迹规划器（chapter6_ 2_ 4. m）的导入变量和导出变量。

同理，将基于 MPC 的路径跟踪控制器也导入 Simulink 中。由于车辆的姿态变化有一点延时，所以在 Simulink/CarSim 联合仿真平台中加入延时模块，如图 6.35 所示。

仿真结果如图 6.36 所示。从图中可以看出，在 3 种不同速度下规划与控制系统都顺利完成了对障碍物的规避以及全局参考轨迹的跟踪。图 6.36（a）中，车辆从原点位置出发，此时并没有感知到障碍物的存在，重规划的轨迹与

图 6.34 S－Function 对话框

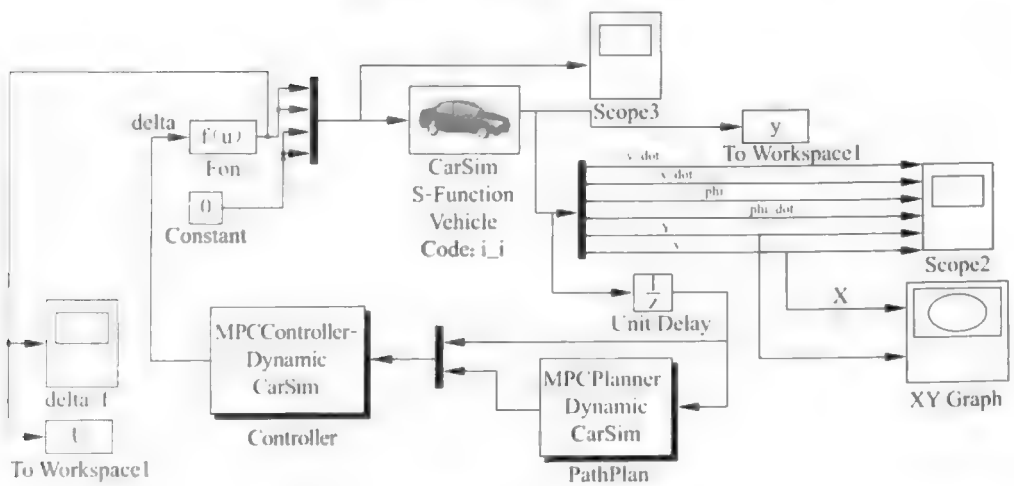


图 6.35 Simulink/CarSim 联合仿真平台

参考轨迹重合。当车辆行驶至 20 m 处时，障碍物信息的加入使得重规划的轨迹偏离了参考轨迹。通过一系列的重规划轨迹，无人驾驶车辆实现了对障碍物的规避，最终跟踪上参考轨迹并保持稳定。图 6.36（b）与图 6.36（c）中，由于车速的增加，相同预测时域内规划出的轨迹区间也相应增加，因此车辆在出发时就能够规划出绕开障碍物的局部轨迹，实现在高速下的障碍物规避与轨迹跟踪。

本章是针对第 4 章中所提出的轨迹跟踪控制系统的不足而进行的补充与完

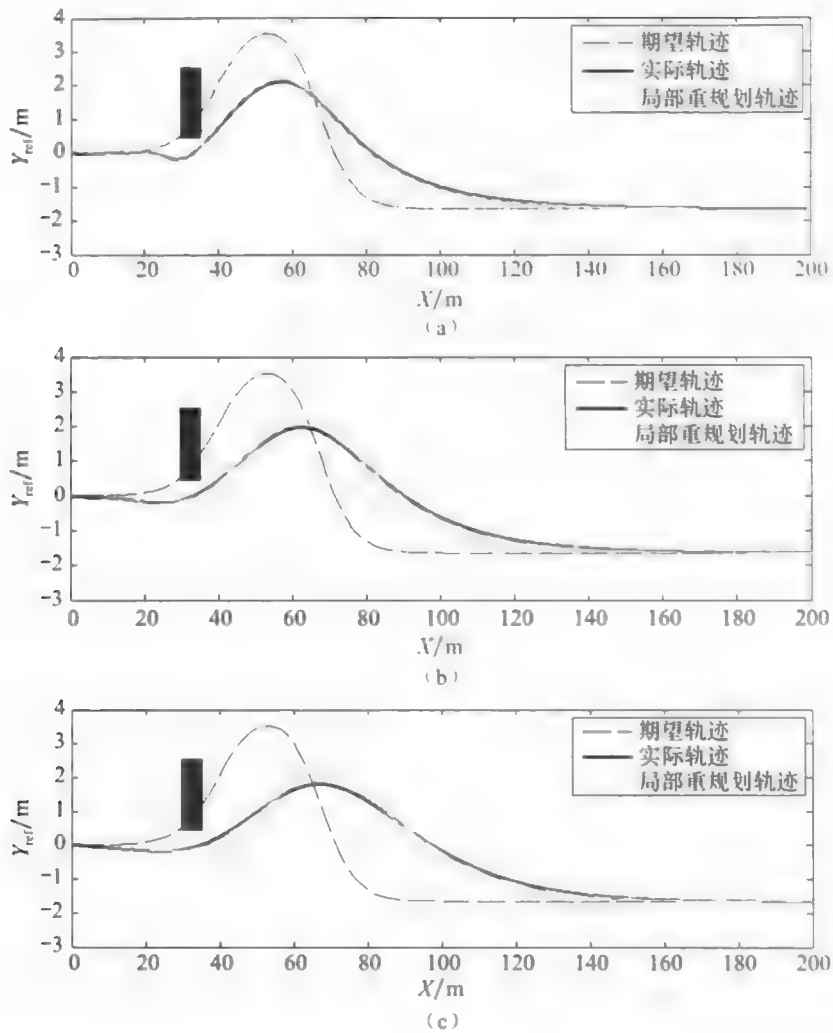


图 6.36　仿真工况 1 的仿真结果

(a) 10 m/s 仿真结果；(b) 20 m/s 仿真结果；(c) 30 m/s 仿真结果

善，在原有的控制层上加入规划层，形成“轨迹重规划+跟踪控制”的双层控制体系。在轨迹重规划算法中，结合非线性模型预测算法与避障功能函数，实现对障碍物的可靠规避。对参考轨迹上轨迹点的选取方法进行了改进，提出了一种融合目标点信息的方法，防止出现车辆倒退的现象。为了简化规划层与控制层之间的传递参数，提出了一种基于 5 次多项式曲线拟合的方法，将规划出的轨迹点进行拟合。统计结果表明，这种方法具有很高的精度。最后，在两种不同的工况下进行仿真实验。结果表明，所设计的双层控制体系能够在不同速度、不同障碍物环境中实现对参考轨迹的可靠、稳定跟踪。

第7章

航向跟踪预估控制算法

模型预测控制虽然能够很好地表达运动规划和跟踪控制过程中模型约束问题，但实际应用时，往往由于计算复杂耗时而无法在嵌入式系统中实现。尽管如此，模型预测控制思想还是可以在无人驾驶车辆控制过程中得到实际应用。本章介绍了著者在实际车辆控制中运用的单步航向预估控制方法，虽然没有进行优化，但在考虑模型约束的基础上，结合PID控制方法，在单片机程序中实现了无人驾驶车辆航向单步预测，取得了较好的控制效果[23]。

7.1 概　　述

航向跟踪是无人驾驶车辆路径跟踪的一种基本方法。在该方法中，位置偏差和航向偏差最后都可以被转化为航向偏差。因此，精确的航向跟踪是实现路径跟踪的前提条件。无人驾驶车辆是一个有较大延迟的高度非线性的复杂系统，建立精确的数学模型十分困难，在进行航向跟踪控制时，参数的变化对系统模型影响较大，其中纵向速度的变化影响最为明显。无人驾驶车辆航向跟踪一般控制方法是把期望航向与车辆实测航向之差作为控制器输入偏差，而控制器输出控制量则为机器人的前轮偏角。

无人驾驶车辆的航向与其纵向速度、横向速度、前轮偏角、车辆绕其重心的转动惯量、重心位置、前后轮侧偏系数以及实际道路情况等诸多因素有关。在常规控制方法中，只考虑了期望航向与实际航向的偏差，而未能包含其他因

素的影响，因此难以达到满意的控制效果。当系统参数，特别是某些敏感参数发生变化时，就必须重新设定控制器参数。例如，用常规 PID 控制器进行航向跟踪实验，在某一纵向速度下整定好 PID 控制参数，而当纵向速度发生很小变化时，就必须重新整定 PID 参数，否则控制性能变坏，超调较大，甚至出现振荡。表现在路径跟踪实验中，则是在一定速度下能较好地完成弯道或急弯等路径跟踪任务，而速度变化后，跟踪误差变大或出现大幅度振荡。因此，在无人驾驶车辆航向跟踪控制中，控制方法应该能对纵向速度等影响因素有一定的自适应能力。

以下首先介绍简化的无人驾驶车辆的二自由度动力学模型，以便对提出的方法进行仿真分析，然后对航向跟踪预估控制算法的基本原理进行说明，最后给出了仿真与实验结果。

7.2 二自由度无人驾驶车辆动力学模型

无人驾驶车辆有纵向、横向和垂直方向的平动，以及侧倾、俯仰和横摆 3 个方向的转动。其中，横向运动和横摆运动基本上是由转向操纵而产生的两种运动。当横向加速度和横摆角速度较小时，常采用经简化的二自由度无人驾驶车辆单轨模型，如图 2.5 所示。将无人驾驶车辆转向机构视为一阶惯性环节，则有：

$$\tau\dot{\delta}_f + \delta_f = \delta_d \tag{7.1}$$

其中，τ 为惯性时间常数；δ_d 为期望前轮偏角，即控制器输出的控制量。

在如图 7.1 所示的无人驾驶车辆路径跟踪示意图中，D 为期望路径上的预瞄点，φ_e 为航向偏差角。假设以下两个条件成立：一是机器人横向速度与纵向速度的比值很小，即纵向速度远大于横向速度；二是航向偏差角 φ_e 很小。再考虑图 7.2 中车辆与路径的相对运动几何关系，有：

$$\dot{\varphi}_e = \omega - v\rho \tag{7.2}$$

$$\dot{y}_e = v\varphi + v_2 \tag{7.3}$$

在航向跟踪控制过程中，可以令道路曲率 $\rho = 1/R$ 为零，即不计实际路径的影响。由式（7.2），有：

$$\dot{\varphi}_e = \omega \tag{7.4}$$

从式（7.4）可以看出，当不计路径影响时，无人驾驶车辆航向变化率即横摆角速度。

以上算式联立即可得出以横摆角速度、横向速度、航向偏差角和前轮偏角

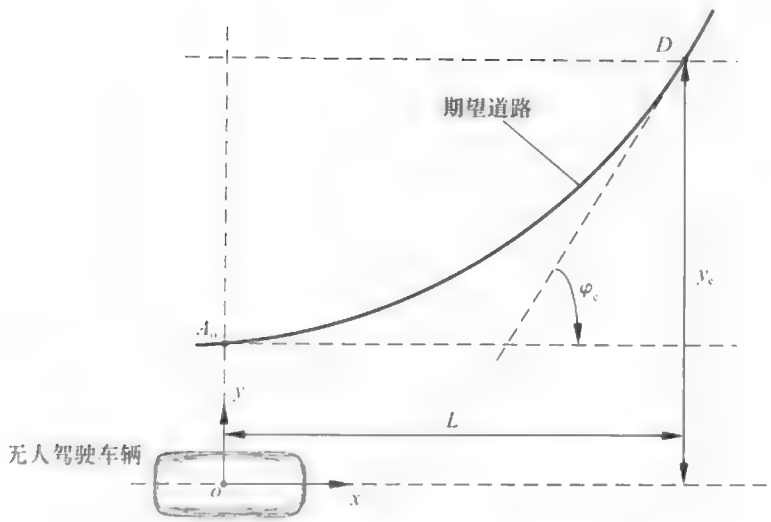


图 7.1 无人驾驶车辆与道路几何关系示意

为状态变量的系统状态空间表达式[23]。

7.3 航向预估算法原理

在实际的航向控制过程中，控制器根据期望航向与无人驾驶车辆实际航向得到航向偏差，再计算控制量，而当执行机构执行这一控制量时，要经过一个采样周期，这时无人驾驶车辆的实际航向已经改变，即控制量执行时已有一个采样周期的滞后，而且采样周期一定时，无人驾驶车辆纵向速度或其他影响因素不同，航向的变化量也不一样。

模型预测控制在预测过程中设置了预测时域，普通的 MPC 控制器的预测输出也是利用下一个采样周期（预测区间）的预测量，本方法也利用航向变化模型在下一周期的输出，可以认为是一种没有经过优化校正的模型预测控制方法。

航向预估算法的基本思想是预测无人驾驶车辆航向变化趋势，并将其计入控制偏差，这样航向变化趋势就可以影响控制器的输出，即无人驾驶车辆的前轮偏角。本书介绍的是一种在简化模型基础上提出的预估算法，其预估模型原理如图 7.2 所示，图中 R 表示无人驾驶车辆绕运动中心点 O 运动的半径。

由式（7.4）可知，无人驾驶车辆在进行航向跟踪控制时，只要求出无人驾驶车辆的横摆角速度，就能得到航向偏差角变化率，而航向偏差角变化率与采样周期的乘积即无人驾驶车辆在一个采样周期内的航向变化量。若设控制器

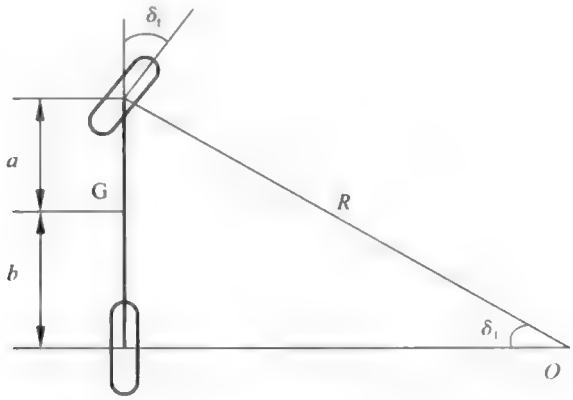


图 7.2　二自由度无人驾驶车辆航向预估模型

采样周期为 T，纵向速度以前轮速度近似，则无人驾驶车辆航向在一个控制周期的变化量 $\Delta\varphi$(rad) 可以近似地计算为

$$\Delta\varphi = vT/R \tag{7.5}$$

前轮绕运动中心的半径 R 可由下式得到：

$$R = (a+b)/\sin(\delta_f) \tag{7.6}$$

将式 (7.6) 代入式 (7.5)，可以得到：

$$\Delta\varphi = vT\sin(\delta_f)/(a+b) \tag{7.7}$$

式中，纵向速度 v 和前轮偏角 δ_f 是在当前采样周期内测得的，在下一个控制周期内会有所变化，但由其变化的连续性计算得到的 $\Delta\varphi$ 作为下一个控制周期内无人驾驶车辆的航向变化估计量是可行的。$\Delta\varphi$ 被称为无人驾驶车辆在下一个控制周期内的航向变化预估量，以下简称航向预估量。在控制算法中，车辆当前航向与航向预估量之和作为航向反馈量，期望航向与航向反馈量的差值作为控制器的输入偏差。航向预估控制框图如图 7.3 所示。

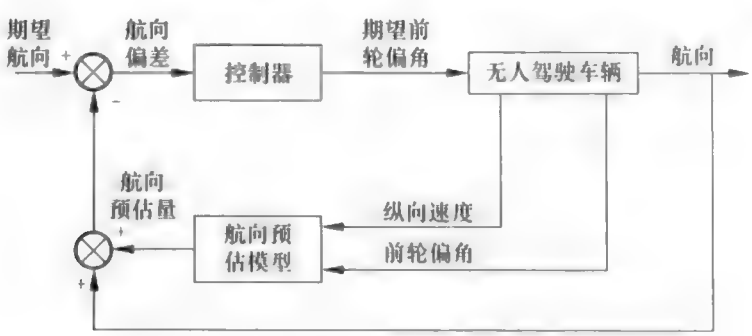


图 7.3　航向跟踪预估控制框图

7.4 PID 控制算法

在仿真和实验中，航向预估控制方法和与之对比的常规控制方法的控制器均采用增量 PID 算法，且两者的比例、积分和微分系数一样。增量 PID 算法如式（7.8）所示：

$$\begin{aligned}\Delta u &= u(k) - u(k-1) \\ &= K_p[e(k) - e(k-1)] + K_i e(k) + K_d[e(k) - 2e(k-1) + e(k-2)]\end{aligned} \tag{7.8}$$

其中 K_p、K_i 和 K_d 分别为比例、积分和微分系数，$u(k)$ 表示第 k（$k=0$，1，2，…）个采样时刻的控制量，$e(k)$ 表示第 k 个采样时刻的航向输入偏差。

7.5 仿真结果

仿真实验在 Matlab/Simulink 图形仿真环境下进行，结构如图 7.4 所示，与实际系统一致，轮式无人驾驶车辆前轮偏角最大设为 35°，因此控制器最大输出绝对位置控制量为 0.611rad，控制器采样周期为 0.064 s，每一控制周期输出增量最大为 0.0224rad。仿真时系统参数选取如表 7.1 所示。进行了航向阶跃响应仿真和路径偏差阶跃响应，以分别验证航向跟踪性能和路径跟踪性能。

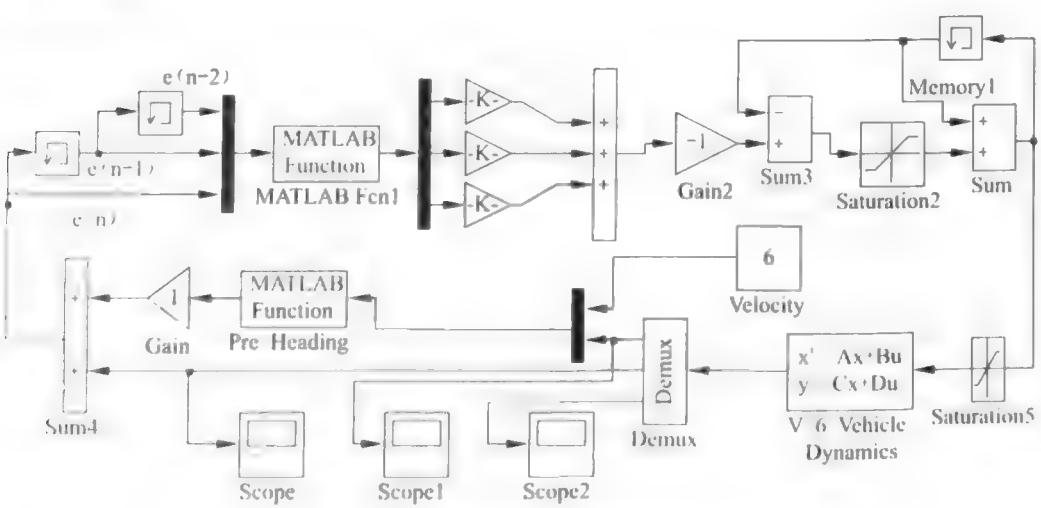


图 7.4 航向跟踪 Matlab/Simulink 图形仿真

表 7.1　航向控制仿真实验系统参数选取

I_z	m	C_1	C_2	a	b	τ
8 890 kg·m^2	3 000 kg	48 000 N/rad	42 000 N/rad	1.56 m	2.0 m	0.5 s

7.5.1　航向阶跃响应仿真

将初始航向偏差设置为 20°（0.35rad），进行航向跟踪阶跃响应实验。仿真时取 2 种纵向速度：4 m/s 和 6 m/s。2 种纵向速度下的 PID 控制系数相同。纵向速度为 4 m/s 有、无航向预估量的航向阶跃响应曲线分别如图 7.5 和图 7.6 所示。纵向速度为 6 m/s 有、无航向预估量航向阶跃响应曲线分别如图 7.7 和图 7.8 所示。

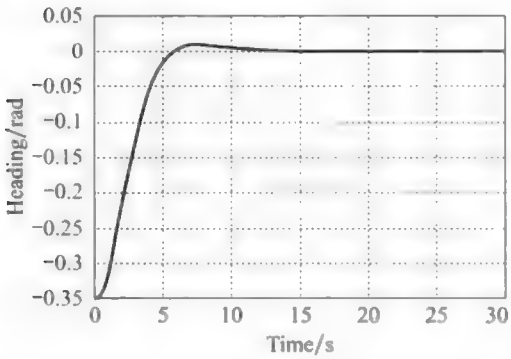


图 7.5　速度为 4 m/s 时有预估量航向阶跃响应

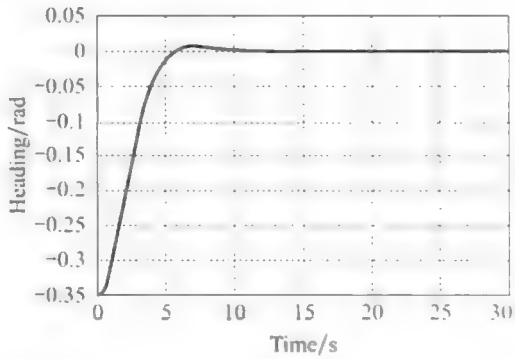


图 7.6　速度为 4 m/s 时无预估量航向阶跃响应

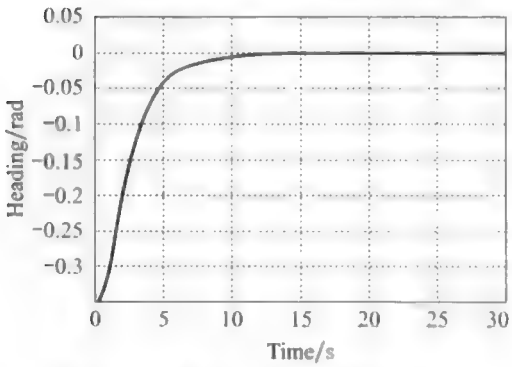


图 7.7　速度为 6 m/s 时有预估量航向阶跃响应

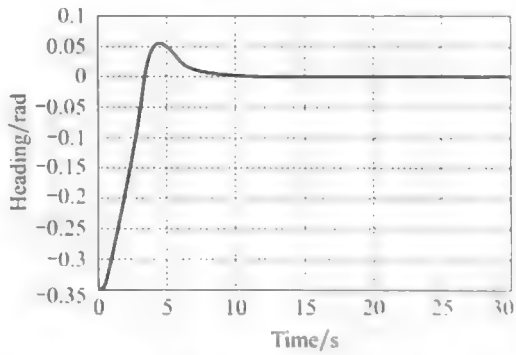


图 7.8　速度为 6 m/s 时无预估量航向阶跃响应

从仿真结果可以看出，纵向速度为 4 m/s 有、无航向预估量时无人驾驶车辆航向阶跃响应曲线基本相同，但当速度变为 6 m/s 时，航向预估控制方法的

阶跃响应曲线较为理想，而用常规控制方法则出现了较大的超调．纵向速度更高时，两种方法的差别更明显．仿真结果说明，有航向预估的 PID 控制器在进行航向跟踪时适应的速度范围较宽，控制系统的鲁棒性比常规控制方法强．

7.5.2 路径偏差阶跃响应仿真

被跟踪路径为一平直道路，无人驾驶车辆的初始路径偏差设置为 -1 m（负值表示无人驾驶车辆位于路径中心的左边），初始航向偏差为0，仿真时也取两种纵向速度：4 m/s 和 6 m/s．两种纵向速度下的 PID 控制系数相同．仿真时对常规方法和航向预估方法的结果进行了对比．速度为 4 m/s 和 6 m/s 时的路径跟踪位置偏差阶跃响应曲线分别如图 7.9 和图 7.10 所示，前轮偏角曲线分别如图 7.11 和图 7.12 所示，航向偏差曲线分别如图 7.13 和图 7.14 所示．

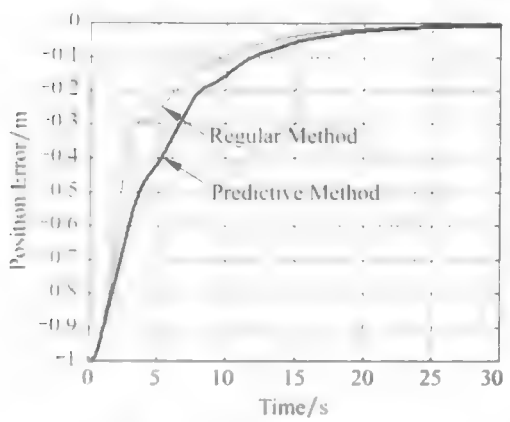


图 7.9 速度为 4 m/s 时路径跟踪阶跃响应

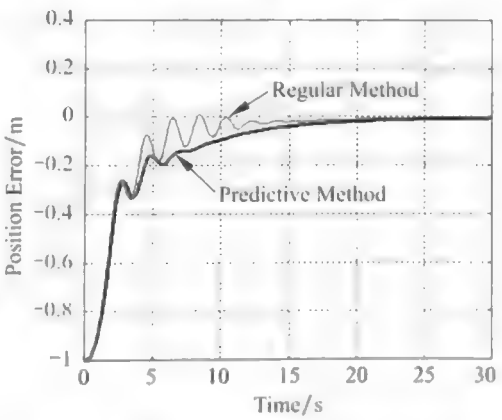


图 7.10 速度为 6 m/s 时路径跟踪阶跃响应

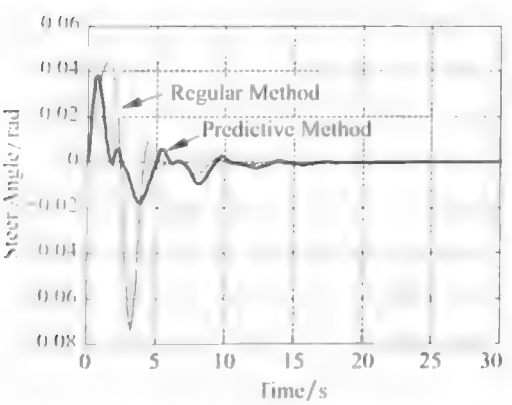


图 7.11 速度为 4 m/s 时前轮偏角曲线

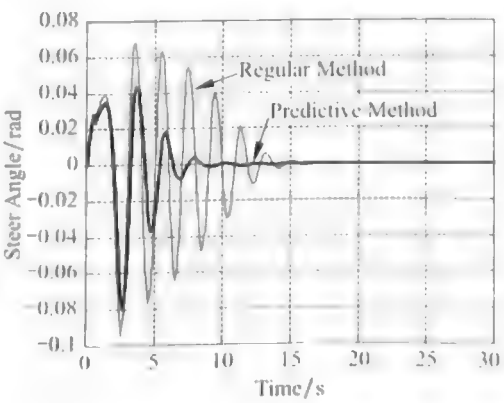


图 7.12 速度为 6 m/s 时前轮偏角曲线

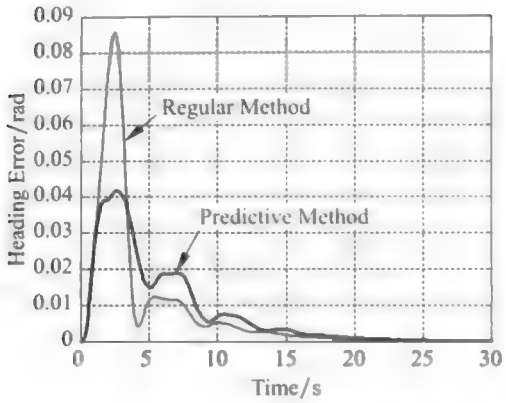


图 7.13　速度为 4 m/s 时航向偏差曲线

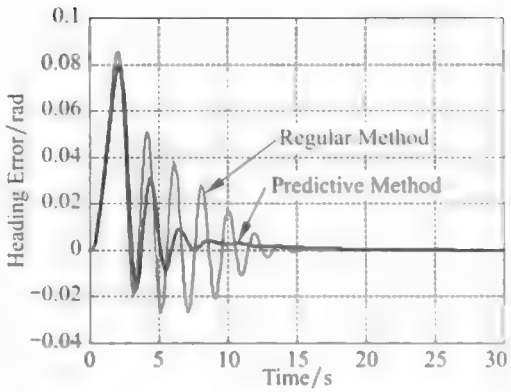


图 7.14　速度为 6 m/s 时航向偏差曲线

从仿真结果可以看出，应用航向预估控制方法的路径跟踪阶跃响应曲线较为平滑，而且当纵向速度变化时，也表现出了较好的适应性；从对控制量前轮偏角的响应曲线和航向偏差曲线来看，航向预估控制方法的稳定性及对纵向速度的适应性明显优于常规方法。

7.6　实验结果

实验平台为一辆具备感知与车辆状态测量功能的无人驾驶车辆。车辆航向由导航系统给出，航向精度为 0.1°。车辆纵向速度由速度控制系统进行控制，实验时能稳定在 ±0.2 m/s 以内。与仿真一样，采用 2 组纵向速度进行：4 m/s 和 6 m/s。实验在平坦开阔的场地进行。实验时所用 PID 参数与仿真实验相同。另外，为了验证航向预估算法在实际路径跟踪控制中的优越性，进行了基于视觉导航的道路跟踪实验。实验过程中纵向速度变化范围较大。

7.6.1　实验 1——航向阶跃实验

开始由无人驾驶车辆跟踪一给定航向，达到期望速度且航向稳定后，给出 20°（0.35rad）的阶跃偏差。纵向速度为 4 m/s 有、无航向预估量的航向阶跃响应曲线分别如图 7.15 和图 7.16 所示。纵向速度为 6 m/s 有、无航向预估量的阶跃响应曲线分别如图 7.17 和图 7.18 所示。

实验结果与仿真结果基本一致。纵向速度为 4 m/s 时，两种方法的阶跃响应曲线基本相同，但预估控制响应曲线较为平滑。纵向速度为 6 m/s 时，预估控制方法控制效果与速度为 4 m/s 时差别不大，但常规控制方法阶跃响应曲线

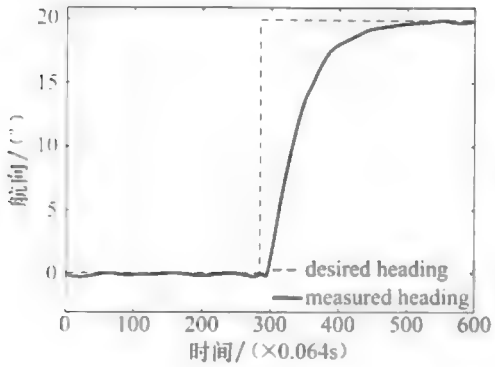


图 7.15 速度为 4 m/s 时有预估量航向阶跃响应

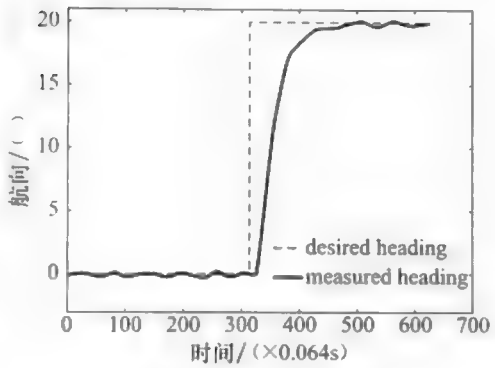


图 7.16 速度为 4 m/s 时无预估量航向阶跃响应

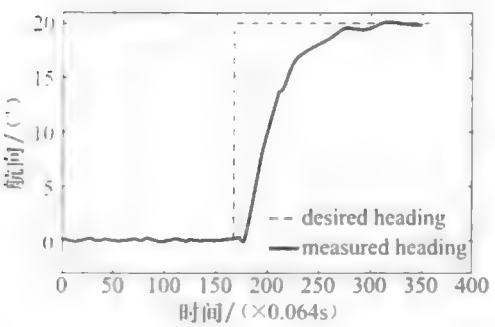


图 7.17 速度为 6 m/s 时有预估量航向阶跃响应

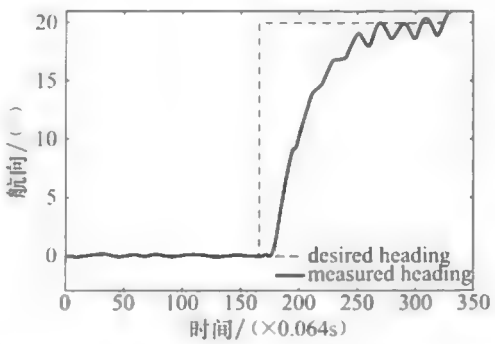


图 7.18 速度为 6 m/s 时无预估量航向阶跃响应

则出现了明显的振荡。同时，从图 7.17 和图 7.18 的直线跟踪阶段曲线（即阶跃响应之前的曲线）可以看出，纵向速度为 4 m/s 时，航向预估控制方法的跟踪曲线较为平滑，速度为 6 m/s 时也有同样的效果。

7.6.2 实验 2——航向连续跟踪实验

实验过程中，每 5 个控制周期（5 × 0.064 s）期望航向减少 1°，减少 20° 后期望航向保持不变。设计本实验的目的是模拟弯道跟踪。实验时纵向速度保持在 6 m/s 有、无预估量的航向响应曲线分别如图 7.19 和图 7.20 所示。

从图中可以看出，航向预估控制方法的响应曲线较为平滑，且无超调，能很好地完成连续跟踪任务；而常规控制方法则出现了大幅振荡，且曲线不平滑。

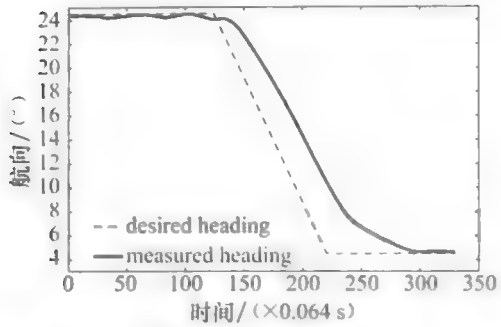


图 7.19　纵向速度为 6 m/s 时有预估量航向连续跟踪响应曲线

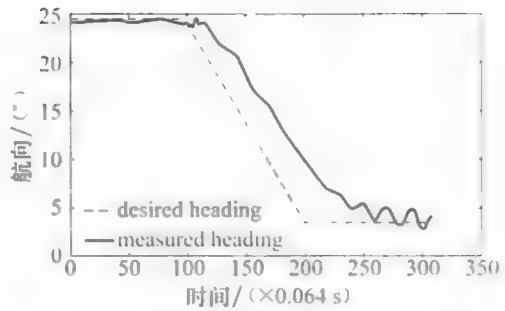


图 7.20　纵向速度为 6 m/s 时无预估量航向连续跟踪响应曲线

从上面的实验可以看出实验结果与仿真结果基本一致，这说明对无人驾驶车辆转向控制的仿真结果是可信的。

7.6.3　实验 3——航向预估算法在路径跟踪控制中的应用

本实验目的是为了验证航向预估算法在路径跟踪控制中的优越性。被跟踪路径由视觉导航系统给出，而局部规划层则根据视觉图像处理结果得到可行驶的路径点，即期望路径。差分 GPS 系统提供的全局坐标精度可达到 0.1 m，通过下面的推算可以得到期望路径坐标点。

实验采用基于航向的路径跟踪方法，预视距离 d 的选取与速度呈线性关系：

$$d = d_0 + k_v v \tag{7.9}$$

其中，d_0 是纵向速度为 0 时的预视距离，被称为基本预视距离；k_v 是纵向速度关联系数，实际选取时不宜过大。

控制器采用第 7.4 节所述的增量 PID 控制算法。为了说明航向预估算法对纵向速度的适应性，在纵向速度为 4 m/s、8 m/s 和 12 m/s 时分别整定了 3 组 PID 参数。实验时无人驾驶车辆纵向速度在 0 ~ 14 m/s 变化。采用如下规则：当纵向速度为 $0 < v_1 \leqslant 6$ m/s 时，PID 控制参数采用 4 m/s 时整定的参数；当纵向速度为 $6 < v_1 \leqslant 10$ m/s 时，PID 控制参数采用 8 m/s 时整定的参数；当纵向速度为 $10 < v_1 \leqslant 14$ m/s 时，PID 控制参数采用 12 m/s 时整定的参数。

无人驾驶车辆全自主行驶的道路跟踪曲线如图 7.21（a）所示，显示的期望路径是根据预视距离选择的目标路径点列。图中的点（0，0）为差分 GPS 基站接收机所处位置。图 7.21（b）是图 7.21（a）中方框部分的放大，图 7.21（c）是期望航向及无人驾驶车辆实际跟踪航向曲线，而图 7.21（d）则是无人驾驶车辆全程的速度曲线。从实验效果进一步证明了航向预估算法对

纵向速度较宽的适应范围。若采用常规 PID 控制，则纵向速度每变化 1 m/s 就必须重新整定 PID 参数，否则难以达到满意的路径跟踪效果。

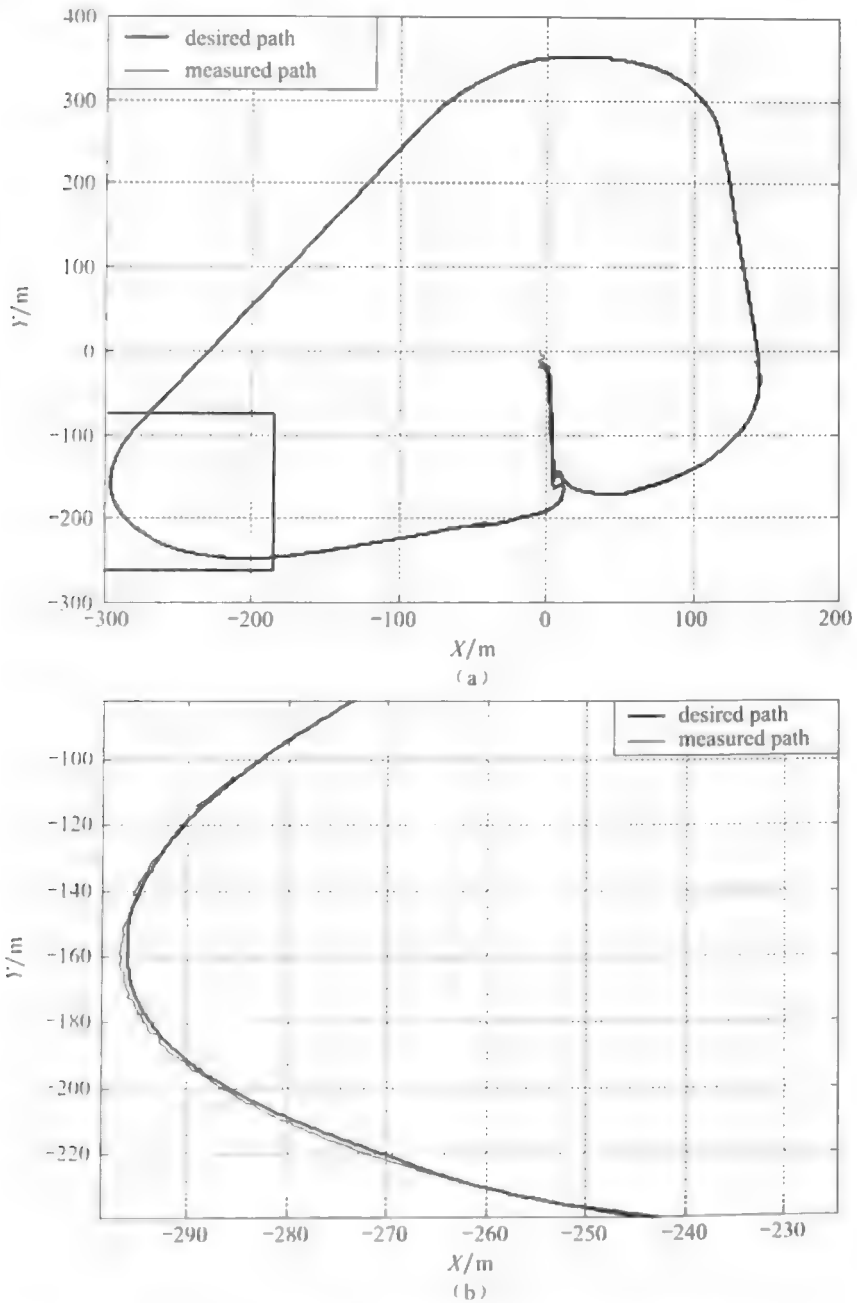
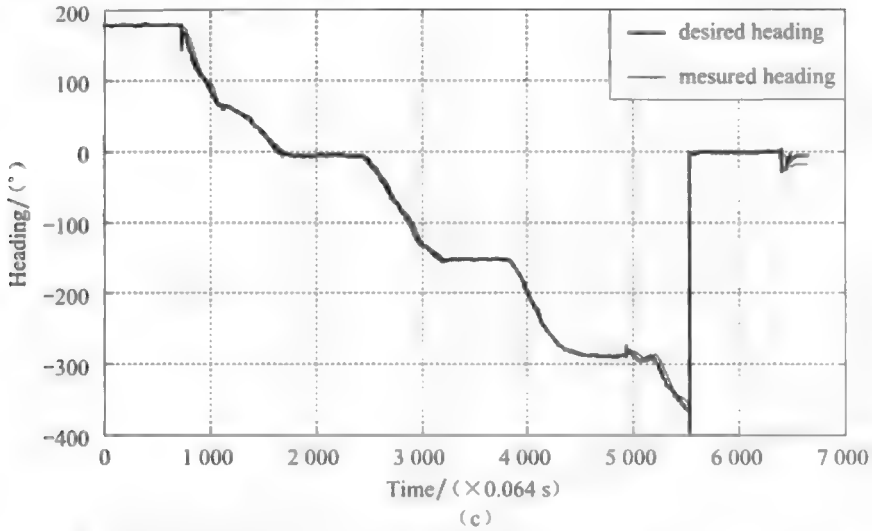
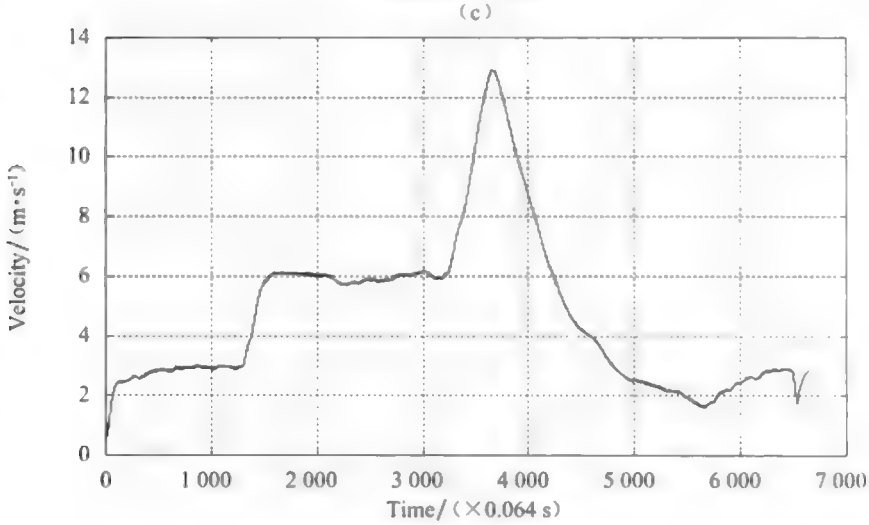


图 7.21 基于视觉导航的无人驾驶车辆道路跟踪实验

(a) 道路跟踪曲线；(b) 图 (a) 中方框部分放大曲线



(c)



(d)

图 7.21　基于视觉导航的无人驾驶车辆道路跟踪实验（续）
（c）航向跟踪曲线；（d）纵向速度曲线

本章提出一种无人驾驶车辆航向跟踪预估控制算法，根据无人驾驶车辆前轮偏角和纵向速度来计算航向的变化量作为航向预估量，并将航向预估量与无人驾驶车辆实际航向之和作为控制的反馈航向。航向预估量对无人驾驶车辆航向变化趋势做了预测。由于在航向预估量中考虑了无人驾驶车辆纵向速度和实时前轮偏角的影响，因而控制器对无人驾驶车辆纵向速度变化的适应范围较宽。算法简单实用，几乎没有添加任何计算量，实时性好。在仿真

实验和实车实验中，把用航向预估算法和 PID 控制算法相结合，进行无人驾驶车辆的航向跟踪和路径跟踪实验。结果表明，与常规方法相比，航向预估算法有效地改善了控制性能，适应的纵向速度范围较常规控制方法要宽，对系统参数变化的敏感性较低，减少了控制器对无人驾驶车辆模型的依赖，提高了控制器的鲁棒性。

附录 A

CarSim 8.02 应用高版本 Matlab

首先确认您的 Matlab 是 64bit 还是 32bit，以下以 Matlab R2012b（64bit）为例：

① 选择 Models：Simulink，如图 A.1 所示。

② 选择红色设置 CarSim 和 Simulink 链接工程文件，如图 A.1 所示

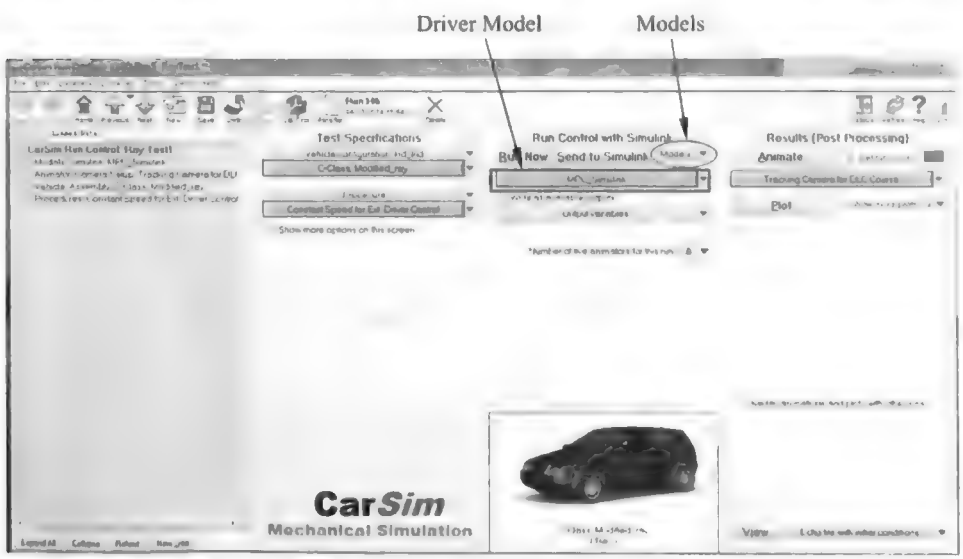


图 A.1

出现 Simulink 设置对话框，如图 A.2 所示；选择 For 64 - bit Windows OS：

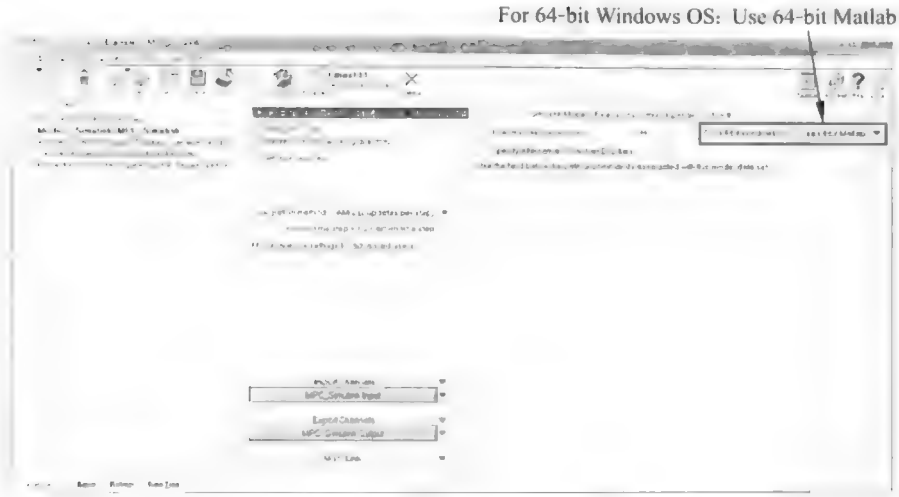


图 A.2

Use 64 - bit Matlab，同时保证 Simulink Model 的后缀为 .mdl，并且路径均为英文。

返回 Matlab 对话框，如图 A.3 所示；选择“SetPath”添加 CarSim 路径下的求解器进 Matlab 路径

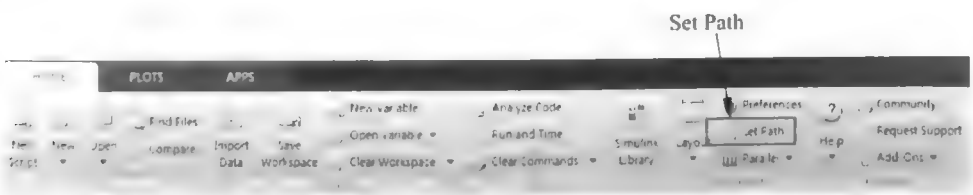


图 A.3

选择“Add with Subfolders”，找到 CarSim 的求解器文件夹，添加到 Matlab 路径中，例如（C:\ProgramFiles（x86）\ CarSim802_ Prog \ Programs \ Solvers）保存设置（Save）如图 A.4 所示

重启电脑，重新打开 Matlab 的 Simulink Library Browser（如图 A.5 所示），CarSim 模块就出现在 Library 里面了

拖入 CarSim 模块进入 Simulink 工程中，保存工程选择文件后缀为 .mdl（CarSim 8.02 不支持后缀为 .slx 的文件和中文路径）

C:\ProgramFiles(x86)\CarSim802_Prog\Programs\Solvers\Default
C:\ProgramFiles(x86)\CarSim802_Prog\Programs\Solvers
C:\ProgramFiles(x86)\CarSim802_Prog\Programs\Solvers\Default64
C:\ProgramFiles(x86)\CarSim802_Prog\Programs\Solvers\ReadMe

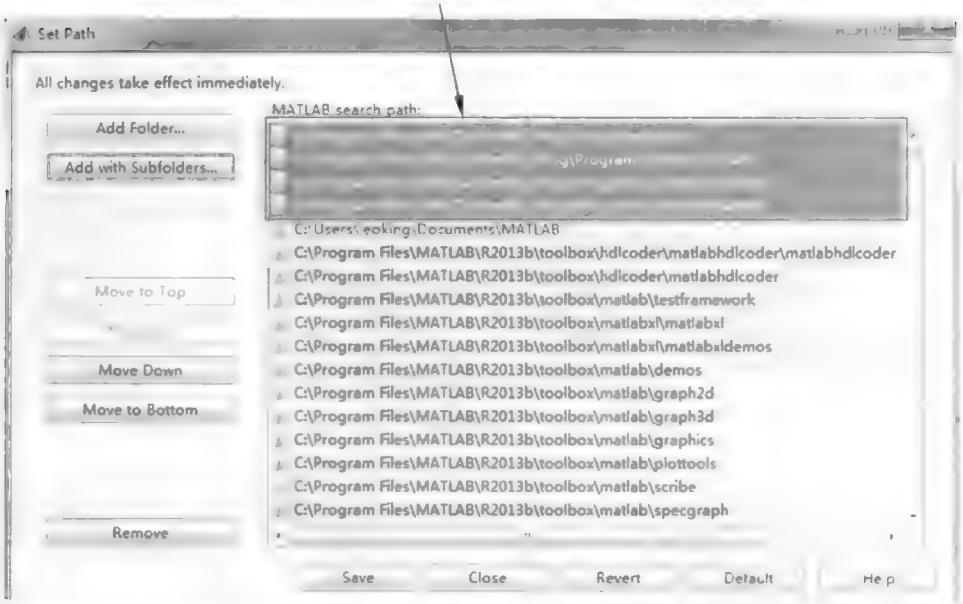


图 A.4

图 A.5

符号表

φ：车体的横摆角（航向角）
δ：轮胎偏转角
δ_f：前轮偏转角
δ_r：后轮偏转角
$\Delta\delta_f$：每个控制周期的前轮偏角增量
α：轮胎侧偏角
α_f：前轮侧偏角
α_r：后轮侧偏角
β：质心侧偏角
v_r：车辆后轴中心速度
v_f：车辆前轴中心速度
v_c：轮胎侧向速度
v_l：轮胎纵向速度
l：轴距
L：车长
W：车宽
a，b：质心到前、后轴的距离
I_z：车辆绕 z 轴的转动惯量
m：车辆整备质量
R：车辆转向半径
r：车轮半径
F_l：轮胎纵向力
F_c：轮胎侧向力
C_l：轮胎纵向刚度
C_c：轮胎侧偏刚度
C_{cf}，C_{cr}：前、后轮胎侧偏刚度

C_{lf}，C_{lr}：前、后轮胎纵向刚度
F_{lf}、F_{lr}：前、后轮胎受到的纵向力
F_{cf}，F_{cr}：前、后轮胎受到的侧向力
F_{xf}，F_{xr}：前、后轮胎受到的 x 方向的力
F_{yf}，F_{yr}：前、后轮胎受到的 y 方向的力
F_z：轮胎所受到的垂向载荷
F_{zf}，F_{zr}：前、后轮胎受到的垂向载荷
M_r：轮胎翻转力矩
M_y：轮胎阻力矩
M_z：轮胎回正力矩
B：魔术模型中曲线的刚度因子
C：魔术模型中曲线的形状因子
D：魔术模型中曲线巅因子
E：魔术模型中曲线的曲率因子
S_h：曲线的水平方向漂移
S_v：曲线的垂直方向漂移
ω：车辆横摆角速度
ω_t：车轮旋转角速度
μ：地面附着系数
s：纵向滑移率
ξ：状态变量
u：控制变量
η：系统输出
ξ_{kin}：运动学模型中的状态量
u_{kin}：运动学模型中的控制量
η_{kin}：运动学模型中的输出量
ξ_{dyn}：动力学模型中的状态量
u_{dyn}：动力学模型中的控制量
η_{dyn}：动力学模型中的输出量
ξ_{pm}：点质量模型中的状态量
u_{pm}：点质量模型中的控制量
T：采样周期
h：龙格－库塔迭代算法中的步长
N：时域（包含预测时域和控制时域）

N_p：预测时域

N_c：控制时域

$\boldsymbol{u}^*(t)$：t 时刻最优控制量

$\boldsymbol{U}^*(t)$：t 时刻最优控制序列

$\boldsymbol{Y}(t)$：t 时刻预测时域内系统输出序列

$\boldsymbol{Y}_{\min}$，$\boldsymbol{Y}_{\min}$：系统输出序列约束

$\boldsymbol{\xi}(t+k\mid t)$：在 t 时刻对 $t+k$ 时刻的预测状态量

$\boldsymbol{\xi}_r$：参考系统的状态变量

$\boldsymbol{u}_r$：参考系统的控制变量

$\Delta\boldsymbol{u}$：控制增量

$\boldsymbol{u}_{\min}$，$\boldsymbol{u}_{\min}$：控制量约束

$\boldsymbol{u}_{\mathrm{kin},\min}$，$\boldsymbol{u}_{\mathrm{kin},\min}$：运动学模型中控制量约束

$\boldsymbol{u}_{\mathrm{dyn},\min}$，$\boldsymbol{u}_{\mathrm{dyn},\min}$：动力学模型中控制量约束

$\Delta\boldsymbol{u}_{\min}$，$\Delta\boldsymbol{u}_{\max}$：控制增量极约束

$\Delta\boldsymbol{u}_{\mathrm{kin},\min}$，$\Delta\boldsymbol{u}_{\mathrm{kin},\max}$：运动学模型中控制增量约束

$\Delta\boldsymbol{u}_{\mathrm{dyn},\min}$，$\Delta\boldsymbol{u}_{\mathrm{dyn},\max}$：动力学模型中控制增量约束

n_x：状态变量维数

m_u：控制变量维数

$\boldsymbol{X}$：状态变量约束

$\boldsymbol{\Gamma}$：控制变量约束

$\boldsymbol{\eta}_{\mathrm{ref}}$：系统参考输出

$\boldsymbol{\eta}_{\mathrm{ref,local}}$：局部参考输出

y_{hc}：硬约束输出

y_{sc}：软约束输出

$\boldsymbol{y}_{\mathrm{hc},\min}$，$\boldsymbol{y}_{\mathrm{hc},\max}$：硬约束输出极限值

$\boldsymbol{y}_{\mathrm{sc},\min}$，$\boldsymbol{y}_{\mathrm{sc},\max}$：软约束输出极限值

$\boldsymbol{e}_t$：预测时域内的跟踪误差

$\boldsymbol{A}_t$，$\boldsymbol{B}_t$：t 时刻的转移矩阵

$\boldsymbol{A}_{k,t}$，$\boldsymbol{B}_{k,t}$：t 时刻对 k 时刻预测的转移矩阵

$\boldsymbol{A}_{\mathrm{kin}}(k)$，$\boldsymbol{B}_{\mathrm{kin}}(k)$：线性运动学模型转移矩阵的离散形式

$\boldsymbol{A}_{\mathrm{dyn}}(t)$，$\boldsymbol{B}_{\mathrm{dyn}}(t)$：$t$ 时刻线性动力学模型的转移矩阵（连续形式）

$\boldsymbol{A}_{\mathrm{dyn}}(k)$，$\boldsymbol{B}_{\mathrm{dyn}}(k)$：线性动力学模型转移矩阵的离散形式

Y_{ref}，φ_{ref}：双移线参考轨迹

a_x：车辆纵向加速度

a_y：车辆侧向加速度

ε：松弛因子

ρ：松弛因子的权重系数

S_{obs}：避障函数的权重系数

ζ：较小的正数

a_p，b_p：5 次曲线拟合中待求系数矩阵

参考文献

1 李波. 人在回路的无人驾驶车辆启发式全局路径规划算法研究 [D]. 北京：北京理工大学，2013.

2 Gong J, Huang W, Xiong G. Genetic Algorithm Based Combinatorial Auction Method for Multi-robot Task Allocation [J]. Journal of Beijing Institute of Technology, 2007, 16 (2): 151 – 156.

3 黄宛宁，龚建伟. 基于改进遗传算法的多机器人任务分配组合拍卖方法 [J]. 计算机仿真，2006，23 (11)：164 – 167.

4 熊光明，龚建伟，段玉林，满益明. 基于行为优先与任务协调的多移动机器人遥操作 [J]. 北京理工大学学报，2007，27 (4)：299 – 302.

5 Hu Y, Gong J, Jiang Y, et al. Hybrid Map-Based Navigation Method for Unmanned Ground Vehicle in Urban Scenario [J]. Remote Sensing, 2013, 5 (8): 3662 – 3680.

6 Gong J, Wang A, Zhai Y, et al. High Speed Lane Recognition Under Complex Road Conditions [C]. IEEE Intelligent Vehicles Symposium, 2008 (6): 566 – 570.

7 Zhou S, Xi J, Gong J, et al. A Novel Lane Detection Based on Geometrical Model and Gabor Filter [C]. IEEE Intelligent Vehicles Symposium, San Diego, CA, 2010 (6): 59 – 64.

8 Zhou S, Gong J, Xiong G, et al. Road Detection Using Support Vector Machine Based on Online Learning and Evaluation [C]. IEEE Intelligent Vehicles Symposium, San Diego, 2010 (6): 256 – 261.

9 龚建伟，叶春玉，姜岩，熊光明. 多层感知器自监督在线学习非结构化道路识别 [J]. 北京理工大学学报，2014，34 (1)：49 – 54.

10 Hu Y, Li X, Gong J. Multi-Feature Extraction for Drivable Road Region Detection with a Two-Dimensional Laser Range Finder [J]. Advanced Materials Research, 2011 (304): 381 – 386.

11 Jiang Y, Zhou S, Gong J, et al. Traffic Sign Recognition Using Ridge Regression and OTSU Method [C]. IEEE Intelligent Vehicles Symposium, 2011 (6): 613 – 618.

12 Gong J, Jiang Y, Xiong G, et al. The Recognition and Tracking of Traffic Lights Based on Color Segmentation and CAMSHIFT for Intelligent Vehicles [C]. IEEE Intelligent Vehicles Symposium, 2010 (6): 431 – 435.

13 关超华，杨泳月，杨慧岩，龚建伟. 基于改进 DBSCAN 算法的激光雷达车辆探测方法 [J]. 北京理工大学学报，2010，30 (6)：732 – 736.

[14] Gong J, Duan Y, Man Y, Xiong G. VPH +: An Enhanced Vector Polar Histogram Method for Mobile Robot Obstacle Avoidance [C]. International Conference on Mechatronics and Automation, 2007 (8): 2784 - 2788.

[15] 张浩杰，龚建伟，姜岩，熊光明，陈慧岩. 基于变维度状态空间的增量启发式路径规划方法研究 [J]. 自动化学报，2013，39 (10)：1602 - 1610.

[16] 姜岩，龚建伟，熊光明，陈慧岩. 基于运动微分约束的无人车辆纵横向协同规划算法的研究 [J]. 自动化学报，2013，39 (12)：2012 - 2020.

[17] Zhang H, Gong J, Jiang Y, Xiong G, Chen H. An Iterative Linear Quadratic Regulator Based Trajectory Tracking Controller for Wheeled Mobile Robot [J]. Journal of Zhejiang University SCIENCE: Computers & Electronics, 2012, 13 (8): 593 - 600.

[18] Yu H, Gong J, Iagnemma K, Jiang Y, Duan J. Robotic Wheeled Vehicle Ripple Tentacles Motion Planning Method [C]. IEEE Intelligent Vehicles Symposium, 2012 (6): 1156 - 1161.

[19] Li B, Gong J, Jiang Y, Nasry H, Xiong G. ARA * +: Improved Path Planning Algorithm Based on ARA * [C]. IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI - IAT), Dec. 4, 2012: 361 - 365.

[20] Zhang H, Xiong G, Su B, Gong J, Jiang Y, Chen H, Lan W. Anytime Path Planning In Graduated State Space [C]. IEEE Intelligent Vehicles Symposium, 2013 (6): 358 - 362.

[21] 张浩杰. 不确定环境下基于启发式搜索的无人驾驶车辆路径规划研究 [D]. 北京：北京理工大学，2013.

[22] Gong J, Duan Y, Liu K, etc. A Robust Multistrategy Unmanned Ground Vehicle Navigation Method Using Laser Radar [C]. IEEE Intelligent Vehicles Symposium, 2009: 417 - 424.

[23] 龚建伟. 移动机器人横向与纵向控制方法研究 [D]. 北京：北京理工大学，2002.

[24] 龚建伟，陆际联，黄文宇. 轮式移动机器人航向跟踪预估控制算法 [J]. 机器人，2001，22 (3)：193 - 196.

[25] Aguiar P, Hespanha P, Kokotovic V. Path-Following for Non-Minimum Phase Systems Removes Performance Limitations [C]. IEEE Transactions on Automatic Control, 2005, 50 (2): 234 - 239.

[26] Hilgert J, Hirsch K, Bertram T, et al. Emergency Path Planning for Autonomous Vehicles Using Elastic Band Theory [C]. IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 2003 (2): 1390 - 1395.

[27] Schouwenaars T, How J, FERON E. Receding Horizon Path Planning with Implicit Safety Guarantees [C]. Proceedings of the American Control Conference, 2004 (1): 5576 - 5583.

[28] Von Hundelshausen F, Himmelsbach M, Hecker F, et al. Driving with Tentacles: Inte-

gral Structures for Sensing and Motion [J]. Journal of Field Robotics, 2008 25 (9): 640 – 673.

29] 张纯刚, 席裕庚. 动态未知环境下移动机器人的滚动路径规划 [J]. 机器人, 2002, 24 (1): 71 – 75.

30 从晋峰, 安向京, 陈虹, 等. 基于滚动优化原理的类车机器人路径跟踪控制 [J]. 吉林大学学报: 工学版, 2012, 42 (1): 182 – 187.

31 Herman A L, Conway B A. Direct Optimization Using Collocation Based on High-order Gauss-Lobatto Quadrature Rules [J]. Journal of Guidance, Control, and Dynamics, 1996, 19 (3): 592 – 599.

32 Bhattacharya R, Balas G J, Kaya M A, et al. Nonlinear Receding Horizon Control of an F – 16 Aircraft [J]. Journal of Guidance, Control, and Dynamics, 2002, 25 (5): 924 – 931.

33] Ross I M, Fahroo F. Pseudospectral Methods for Optimal Motion Planning of Differentially Flat Systems [J]. IEEE Transactions on Automatic Control, 2004, 49 (8): 1410 – 1413.

34 程英英. 基于微分平坦的轮式移动机器人轨迹规划 [D]. 长春: 吉林大学, 2008.

35] 马东扬. 基于滚动优化的轮式移动机器人轨迹规划 [D]. 长春: 吉林大学, 2009.

36 Tahirovic A, Magnani G. Passivity-based Model Predictive Control for Mobile Robot Navigation Planning in Rough Terrains [C]. 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010: 307 – 312.

37 Wang D, Qi F. Trajectory Planning for a Four-wheel-steering Vehicle [C]. IEEE International Conference on Robotics and Automation, 2001: 3320 – 3325.

38] Muller B, Deutscher J, Grodde S. Continuous Curvature Trajectory Design and Feedforward Control for Parking a Car [J]. IEEE Transactions on Control Systems Technology, 2007, 15 (3): 541 – 553.

39, 宋金泽, 戴斌, 单恩忠, 等. 融合动力学约束的自主平行泊车轨迹生成方法 [J]. 中南大学学报: 自然科学版, 2009, 40 (1), 135 – 141.

40 Malik J. High-quality Vehicle Trajectory Generation from Video Data Based on Vehicle Detection and Description [J]. Proceedings IEEE Intelligent Transportation Systems, 2003 (1): 176 – 182.

41 Velenis E, Tsiotras P. Optimal Velocity Profile Generation for Given Acceleration Limits: The Half-car Model Case [C]. IEEE International Symposium on Industrial Electronics, 2005 (1): 1478 – 1483.

42 Spenko M, Kuroda Y, Dubowsky S, et al. Hazard Avoidance for High-speed Mobile Robots in Rough Terrain [J]. Journal of Field Robotics, 2006, 23 (5): 311 – 331.

43] Gao T, Lin T, Borrelli F, et al. Predictive Control of Autonomous Ground Vehicles with Obstacle Avoidance on Slippery Roads [C]. Proceedings of the ASME 2010 Dynamic Sys-

tems and Control Conference, 2010: 265 - 272.

[44] Bryson A. E., Ho Y. C. Applied Optimal Control: Optimization, Estimation, and Control [M]. Waltham, MA: Blaisdell, 1969.

[45] Thrun S, Montemerlo T, et al. Stanley: the Robot that Won the DARPA Grand Challenge [J]. Journal of Field Robotics, 2006, 23 (9): 661 - 692.

[46] Johansson B, Gafvert M. Untripped SUV Rollover Detection and Prevention [J]. Proc. 43rd IEEE Conf. Decision and Control, 2004 (5): 5461 - 5466.

[47] Falcone P, Borrelli F, Asgari J, et al. Predictive Active Steering Control for Autonomous Vehicle Systems [J]. IEEE Transactions on Control Systems Technology, 2007, 15 (3): 566 - 580.

[48] Velenis E, Frazzoli E, Tsiotras P. Steady-state Cornering Equilibria and Stabilization for a Vehicle During Extreme Operating Condition [J]. International Journal of Vehicle Autonomous Systems, 2010, 8 (2/3/4): 217 - 241.

[49] Liu J, Jayakumar P, Overholt J L, et al. The Role of Model Fidelity in Model Predictive Control Based Hazard Avoidance in Unmanned Ground Vehicles Using LIDAR Sensors [C]. Proceedings of the ASME Dynamic Systems and Control Conference, 2013: 1 - 10.

[50] Kuhne F, Lages WF, et al. Model Predictive Control of a Mobile Robot Using Linearization [J]. Prodeedings of Mechatronics and Robototics, 2004 (4): 525 - 530.

[51] Peters S, Frazzoli E, Iagnemma K. Differential Flatness of a Front-steered Vehicle with Tire Force Control [C]. Proceedings of the IEEE International Conference on Robots and Systems, 2011: 298 - 304.

[52] Spenko M, Kuroda Y, Dubowsky S, Iagnemma K. Hazard Avoidance for High-speed Mobile Robots in Rough Terrain [J]. Journal of Field Robotics, 2006, 23 (5): 311 - 331.

[53] Hattori Y, Ono E, Hosoe S. Optimum Vehicle Trajectory Control for Obstacle Avoidance Problem [J]. IEEE Transactions on Mechatronics, 2006, 11 (5): 507 - 512.

[54] Beal C E, Gerdes J C. Model Predictive Control for Vehicle Stabilization at the Limits of Handling [J]. IEEE Transactions on Control Systems Technology, 2012: 1 - 12.

[55] Lee J H, Yoo W S. An Improved Model-based Predictive Control of Vehicle Trajectory by Using Nonlinear Function [J]. Journal of Mechanical Science and Technology, 2009, 23 (4): 918 - 922.

[56] Fliess M, Levine J L, Martin P, Rouchon P. Flatness and Defect of Non-linear Systems: Introductory Theory and Examples [J]. International Journal of Control, 1995, 61 (6): 1327 - 1361.

[57] Peters S, Bobrow J, Iagnemma K. Stabilizing a Vehicle Near Rollover: an Analogy to Cart-Pole Stabilization [C]. IEEE International Conference on Robotics and Automation, 2010 (5): 5194 - 5200.

[58] Velenis E, Frazzoli E, Tsiotras P. Steady-state Cornering Equilibria and Stabilization for a Vehicle During Extreme Operating Condition [J]. International Journal of Vehicle Autonomous Systems, 2010, 8 (2/3/4): 217 - 241.

[59] Yildirim E A, Wright S J. Warm-start Strategies in Interior-point Methods for Linear Programming [J]. SIAM Journal on Optimization, 2002, 12 (3): 782 - 810.

[60] Yang W, Boyd S. Fast Model Predictive Control Using Online Optimization [J]. Control Systems Technology, IEEE Transactions, 2010, 18 (2): 267 - 278.

[61] Tondel P, Johansen T A, Bemporad A. An Algorithm for Multi-parametric Quadratic Programming and Explicit MPC Solutions [J]. Automatica, 2003, 39 (3): 489 - 497.

[62] 张聚，王万良. 基于动态规划的约束优化问题多参数规划求解方法及应用 [J]. 控制理论与应用，2008，25 (6)：1135 - 1138.

[63] Shimoda S, Kuroda Y, Iagnemma K. High-speed Navigation of Unmanned Ground Vehicles on Uneven Terrain Using Potential Fields [J]. Robotica, 2007, 25 (4): 409 - 424.

[64] Rajamani R. Vehicle Dynamics and Control [M]. Springer, 2006.

[65] Pacejka H. Tyre and Vehicle Dynamics [M]. Butterworth-Heinemann, 2006.

[66] 席裕庚. 预测控制 [M]. 北京：国防科技图书出版社，1991.

[67] 任慧荣. 类车移动机器人轨迹跟踪控制方法研究 [D]. 天津：天津大学，2008.

[68] 韩光信，陈虹，马苗苗，等. 约束非完整移动机器人轨迹跟踪的非线性预测控制 [J]. 吉林大学学报：工学版，2009，39 (1)：177 - 181.

[69] 曾志文，卢惠民，张辉，等. 基于模型预测控制的移动机器人轨迹跟踪 [J]. 控制工程，2011，(S1)：80 - 85.

[70] Falcone P. Nonlinear Model Predictive Control for Autonomous Vehicles [D]. Benevento: Universitàdel Sannio, 2007.

[71] Borrelli F, Falcone P. MPC-based Approach to Active Steering for Autonomous Vehicle Systems [J]. International Journal of Vehicle Autonomous Systems, 2005, 4 (3): 265 - 291.

[72] 李升波，王建强，李克强. 软约束线性模型预测控制系统的稳定性方法 [J]. 清华大学学报：自然科学版，2010 (11)：1848 - 1852.

[73] Van Zanten A T, Erhardt R, Landesfeind K, et al. VDC Systems Development and Perspective [J]. SAE Transactions, 1998, 107 (6): 424 - 444.

[74] Van Zanten A T, Erhardt R, Lutz A, et al. Simulation for the Development of the Bosch-VDC [J]. SAE Transactions, 1996, 105 (6): 544 - 554.

[75] 宗长富，郭孔辉. 汽车操纵稳定性的客观定量评价指标 [J]. 吉林工业大学自然科学学报，2000，30 (1)：1 - 6.

[76] 熊璐，余卓平，姜炜，等. 基于纵向力分配的轮边驱动电动汽车稳定性控制 [J]. 同济大学学报：自然科学版，2010，38 (3)：417 - 421.

[77] Falcone P, Eric Tseng H, Borrelli F, et al. MPC-based Yaw and Lateral Stabilization via Active Front Steering and Braking [J]. Vehicle System Dynamics, 2008, 46 (S1): 611 -628.

[78] Lee S H, Lee Y O, Kim B A, et al. Proximate Model Predictive Control Strategy for Autonomous Vehicle Lateral Control [C]. American Control Conference (ACC), IEEE, 2012: 3605 -3610.

[79] Kim B A, Lee S H, Lee Y O, et al. Comparative Study of Approximate, Proximate, and Fast Model Predictive Control with Applications to Autonomous Vehicles [C]. 12th International Conference on Control, Automation and Systems (ICCAS), IEEE, 2012: 479 -484.

[80] Eklund J M, Sprinkle J, Sastry S S. Switched and Symmetric Pursuit/Evasion Games Using Online Model Predictive Control with Application to Autonomous Aircraft [J]. Transactions on Control Systems Technology, 2012, 20 (3): 604 -620.

[81] 徐威. 基于模型预测控制的智能车辆运动规划与控制算法研究 [D]. 北京: 北京理工大学, 2014.

[82] 龚建伟, Iagnemma K, 姜岩, 等. 高速地面车辆主动危险规避最优运动规划与控制的动力学模型分析. 国家自然科学基金项目 (51275041) 申报书, 2012.

[83] Gong J, Xu W, Jiang Y. Multi-constrained Model Predictive Control for Autonomous Ground Vehicle Trajectory Tracking [J]. Journal of Beijing Institute of Technology, 2014. (录用).